Phoneme Recognition Using Neural Network and Sequence Learning Model

A thesis presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Master of Science

Yiming Huang

March 2009

This thesis titled

Phoneme Recognition Using Neural Nework And Sequence Learning Model

by

YIMING HUANG

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

_____

Janusz A. Starzyk

Professor of Electrical Engineering and Computer Science

_____

Dennis Irwin

Dean, Russ College of Engineering and Technology

ABSTRACT

HUANG, YIMING, M.S., March 2009, Electrical Engineering

Phoneme Recognition Using Neural Network and Sequence Learning Model (82 pp.)

Director of Thesis: Janusz A. Starzyk

The purpose of this thesis is to describe a biologically motivated approach for phoneme recognition by using a self-organized neural network and sequence learning algorithm. Phoneme recognition in continuous speech is a tough task with a low accuracy rate. By using the sequence learning algorithm to add sequential information of individual phonemes, recognition performance can be improved.

This thesis includes three parts. A self-organized neural network is the first stage, which classifies the input sound waves into forty two different phoneme categories. The 42 output neurons of the neural network are sent to the Sequence Learning block which is composed of Long Term Memory cells. Finally each LTM cell sends a unique feedback strength signal to each output of the neural network to predict the next phoneme, hence, to improve the phoneme recognition based on the sequential information.

Approved: _____

Janusz A. Starzyk

Professor of Electrical Engineering and Computer Science

ACKNOWLEDGMENTS

I would like to take this opportunity to thank all the people who helped and supported me in completing this thesis. Without their assistances and support, my thesis could not have been accomplished.

First and most importantly, I would like give appreciation to my advisor, Professor Janusz A. Starzyk for advising my thesis with his outstanding knowledge and countless time. His working attitude and innovative ideas in research has been of great value for me. I have learned a lot more than just knowledge, but a great work and studying habits.

I also would like to thank my committee members: Professor Jeffery Dill, Professor Maarten Uijt de Haag, and Professor Xiaozhuo Chen for their time to reviewing my thesis and providing useful suggestions.

In these 3 years of graduate study, my lab mates: Haibo, Yinyin, Lily, Xinming and James have provided me a lot of valuable suggestions for research and study. As friends, they also helped me and give me courage and strength in the daily life. So they also deserve my many thanks for all the help they have provided to me.

And last, I would like to thank my parents for their continuous supports. And most importantly, I would like to thank my lovely wife, Chun, for her years of love and support.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

1. INTRODUCTION

1.1 Background

Speech recognition is a very popular research goal in the field of machine intelligence. There are many reasons for automatic speech recognition being widely developed by engineers and scientists around the world. Human-machine interaction is one of the most important reasons. We always dream of ordering machines such as the TV to turn itself on and change channels per our orders, thermostats to adjusting the temperature by themselves to adapt to a human's preferences, or even a robot babysitter to do all the house tasks fast and efficiently. The basic sensory stages of the human-machine interaction are vision recognition and speech recognition. Voice recognition, which is a special kind of speech recognition, is widely used in high security locations. Due to the high demand in the current market, many corporations have already built some Automatic Speech Recognition (ASR) systems: like the dictation system used by IBM and the telephone transaction system used by T-Mobile, AT&T and Philips. Although these systems have been used in commercial area for years already, they still have many problems. First, these systems can only accomplish limited tasks such as recognizing numbers from 0 to 9, or isolated commands (e.g. transfer to customer service, balance request, pay bill, and etc.). Second, they all lack robustness, i.e. these systems have very poor performance in a noisy environment. Some of the "smart" recognition systems can recognize a word, a sentence or even a paragraph but require to be adapted to every new

user, so every new user needs to train the system to recognize his/her specific voice. This approach is not suitable or feasible for a commercial use. These problems lead researchers and scientists to improve the speech recognition systems.

There are three speech recognition technologies that have been developed over the years:

1. Dynamic time warping: an algorithm for measuring similarity between two sequences which may vary in time or speed. However, this technology has been displaced by the more accurate Hidden Markov Model (HMM).
2. Hidden Markov Model: a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters. This algorithm is often used due to its simplicity and feasibility of use.
3. Neural Network based approach: an algorithm for training the system to recognize speech using an artificial neural network. This technology is capable of solving much more complicated recognition tasks, and can handle low quality, noisy data, and speaker independence. If properly developed and used it may be more accurate than HMM.

Although, the Hidden Markov Model is the most popular method used in the commercial speech recognition field due to its simplicity and feasibility of use, its drawbacks motivate many researchers to focus on neural network based approaches. HMM relies highly on the accuracy of the model phonemes and is state dependent. If the nature of the speech is not the same as the given sample or the next phoneme in the word depends on

more than just the previous state, then the recognition rate drops dramatically. In particular, the HMM model cannot properly represent the context of the processed speech, which is an important property of human speech recognition. Another drawback of HMM products is that they are speaker-dependent. The system needs to be trained to create templates of the phonemes and words for each user, i.e. before any user starts to use the recognition system, he/she always needs to train the system with a number of sample words which contains all the phonemes by repeatedly speaking these samples and representing them to the system, then the system can recognize his/her speech by calculating the probability of the current phoneme compared to the database models. This training stage is time consuming. The third disadvantage is that the HMM model always drops the low probability word transitions although they may contain the correct information. [Kom 1998]

A Neural Network (NN) based approach may present a solution to the above drawbacks. Unlike HMM, the NN approach does not require template creation and to a large degree is speaker independent. The recognition system only needs to be trained once to generate the structure of the network. So the total training time for the system is significantly reduced comparing to the HMM. In the neural network system, all the output neurons, where one neuron represents one recognition category, fire at a certain excitation level all the time, so the system does not lose any useful information even for incorrect outputs. In addition, since NNs are compatible with neural based associative memory structures, they are more likely to use speech context in speech recognition. Previously activated words

or concepts may be reused to help recognize a new word or sentence and may remove ambiguity from understanding similarly sounding words. In addition, using speech context will help to overcome the effect of noise or speech interference in case of cocktail party speech recognition.

In general, at the present there is no such speech recognition system that is accurate and robust to all conditions and applications in real world speech based communications. Researchers are looking for a way to simulate human hearing perception, but no one yet is close enough to human ability to understand the speech signals.

1.2 Research Goal

The goal of this master thesis is to find a method to reach an accuracy level of recognition comparable or exceeding those achievable using HMM-based speech recognition systems without too much training, and to provide methods that are largely speaker independent, so that the system can recognize most variations in pronunciation that would be understood by a native speaker of the language. [Kom 1998] In my thesis, the neural network based approach is used for phoneme recognition where the feedback correction method uses the sequence learning model to improve the neural network performance.

There are three stages in my phoneme recognition system. Before the beginning of the first stage, the sound waves need to be preprocessed for the input to the neural network.

There are several ways to preprocess the input sound wave: Feature Extraction, Fourier Transform, Wavelet Transform, etc. Feature Extraction is the most popular method for speech processing but it is possible to produce errors when estimating speech parameters [Wha 1988]; Short Time Fourier Transforms (STFTs) and Linear Predictive Coding (LPC) techniques assume signal stationarity within a given time frame and may therefore lack the ability to accurately analyze localized events [Lon 1996]. In addition, since it contains 1-D information only it is not sufficient for analysis of complex speech signals; Wavelet Transforms can overcome some of the above limitations, and translate a 1-D sound wave into a 2-D image which contains both time and frequency information, they are easy to simulate in a mathematical model. So instead of the popular feature extraction, I used a simpler Wavelet Transform method to obtain the input images. It is not the first time that the Wavelet Transform method applied in this thesis is used in Automatic Speech Recognition. It has been applied with some success in pitch detection, formant tracking and phoneme classification [Lon 1996], due to its high time-frequency resolution [Jan 1996]. In order to limit the size of the inputs, sampling is used to reduce the resolution of the image, and overlapping windows are used to avoid loss of the sampled information.

After preprocessing, these images are input into the neural network to train it for recognition of English syllables or phonemes with the weighted least square approach. In this stage, the outputs are organized into different categories: 42 different phonemes, which are used as the input to the next stage. After this first level recognition of

phonemes by the neural network, a certain recognition rate level is obtained, but it is not accurate enough, so an additional method or system needs to be applied in order to improve the phoneme recognition rate.

The second stage of my simulation model is to create a sequence learning block composed by a number of Long Term Memory (LTM) cells, which saves the sample words (the phoneme sequences) in the sample sentences, using the outputs of the neural network and the desired outputs of this sequence learning block. These LTM cells are created using the sequence learning method based on the architecture introduced by J. Starzyk and H. He in [Sta 2007] for recognizing English words from the syllables or phonemes, and the output of this block is the winner of the LTM cells which has the strongest excitation strength, hence provides the result for word level recognition. However, our goal is to improve the phoneme recognition rate, so the final stage is added to improve the phoneme recognition rate which sends back the word level recognition results to the outputs of the phoneme recognition neural network.

In the final stage of our simulated system, we create feedback signals from the output of each LTM cell and send them back to the neural network outputs to influence the recognition result of the next playing phoneme. The neural network outputs supported by the sequence information should have a more accurate recognition rate. This is because only certain phonemes can compose a word, and by recognizing the word the next

phoneme in the word can be predicted; hence, the performance of the phoneme recognition becomes higher than without the feedback signals.

1.3 Thesis Organization

This thesis is composed of five chapters. Chapter 1 contains the introduction and the organization of the thesis and the background of the current speech recognition research.

Chapter 2 describes the sound signal preprocessing methods and the neural network structure with the weighted least square approach. In this chapter, the structure of a feed-forward network is described in detail and the weighted least square method is presented. The analysis results are also presented in this chapter.

Chapter 3 introduces the sequence learning method and describes in details of the structure of the Long Term Memory (LTM) cells. It also contains the analysis results for word recognition.

Chapter 4 provides information about the phoneme correction method to improve the phoneme recognition rate. The basic idea of this correction method is adding feedback links from the outputs of the LTM cells to the outputs of the phoneme recognition neural network, so the results of word recognition can help the phoneme recognition.

Chapter 5 presents the conclusion of this thesis. This chapter summarizes the work done and the results presented in this thesis. It also provides the work that needs to be done for the future improvement of the speech recognition system.

2. SOUND PREPROCESSING AND FEED-FORWARD NEURAL NETWORK WITH WEIGHTED LEAST SQUARE APPROACH

2.1 Sound Wave Preprocessing

Before the sound waves are sent into the recognition system, they need to be preprocessed to extract as much information as possible. There are several methods to preprocess the sound waves as described above in section 1.2. In this chapter, the Continuous Wavelet Transform is used to extract information from the input sound waves. The processed inputs are the wavelet coefficients obtained by transforming Daubechies wavelet [Dau 1992] with the order of 32. The wavelet coefficients of the signal S at scale *a* and position *b* are calculated by:

$$C_{a,b} = \int_R S(t) \frac{1}{\sqrt{a}} \overline{\Psi\left(\frac{t-b}{a}\right)} dt \quad , \qquad (2.1)$$

where S is the discrete sound wave signal and Ψ is the wavelet.

The inputs to the neural network of our recognition system are composed of these coefficients, and the size of the input matrix is 32 by (the time duration of the current sentence in ms). In order to limit the number of inputs to the neural network and shorten the processing time, one training sentence is transformed at a time. A sampling method was also used to further reduce the resolution of the input image. Before sampling the information in the time domain, a Butterworth 1st order low-pass filter with a cut-off

frequency of 0.02 is applied to take out some of the high frequency noise [Woo 2001]. Then, by setting the sampling limits all the sentences, all transformed phonemes are represented as a 32x32 matrix. The following 3 figures show the sound preprocessing steps:

Figure 1 displays the image of the phoneme “sh” after the Daubechies wavelet transform, the x-axis represents the time duration of this phoneme which is about 1500ms and the y-axis represents the frequency information of the sound at the dedicated time (since we use Daubechies wavelet of the 32^{nd} order, the frequency axis has 32 levels). In this figure the pixels with brighter color mean greater energy at this frequency and this time, and the pixels with darker color means less energy. Figure 2 shows the transformed data after passing through a Butterworth low-pass filter which filtered out some high frequency noise for optimal processing. Then the filtered data gets sampled by a digital sampler and reduced to the size of the image to 32x32 for faster and more efficient analysis. The sampled image is shown in Figure 3. As we can see, the final sampled image is still similar to the original image of wavelet transform, so the most important information from the sound wave has been preserved.

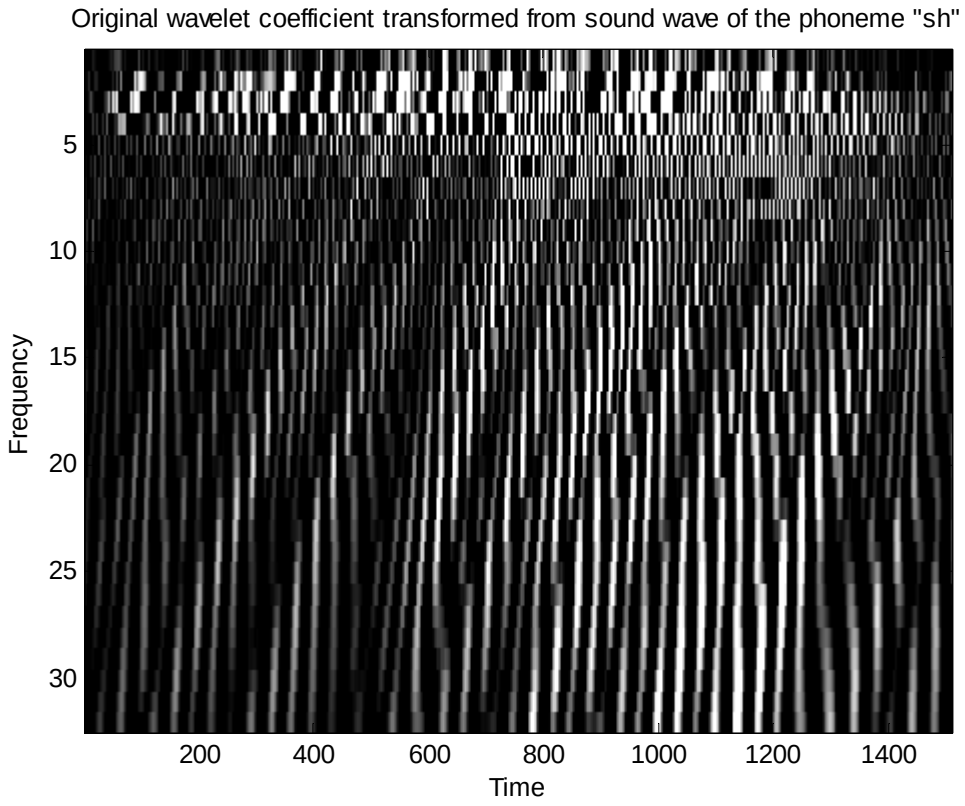


Figure 1. The image of original wavelet coefficient transformed from sound wave of the phoneme “sh”.

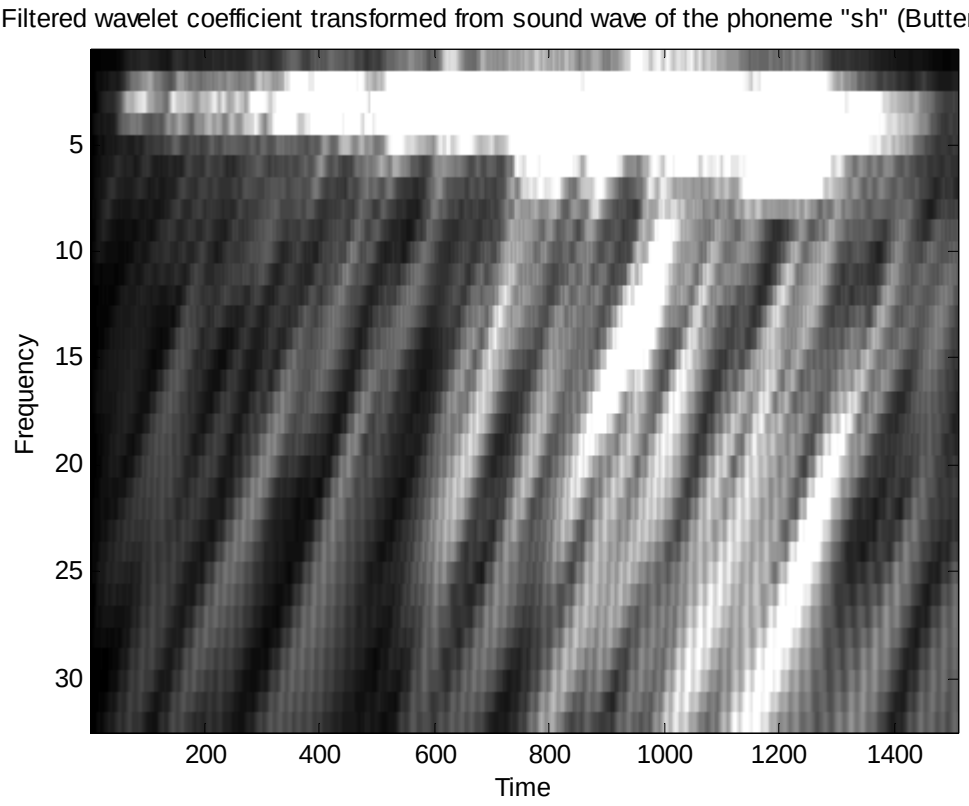


Figure 2. The image of filtered wavelet coefficient transformed from sound wave of the phoneme “sh” using Butterworth low-pass filter.

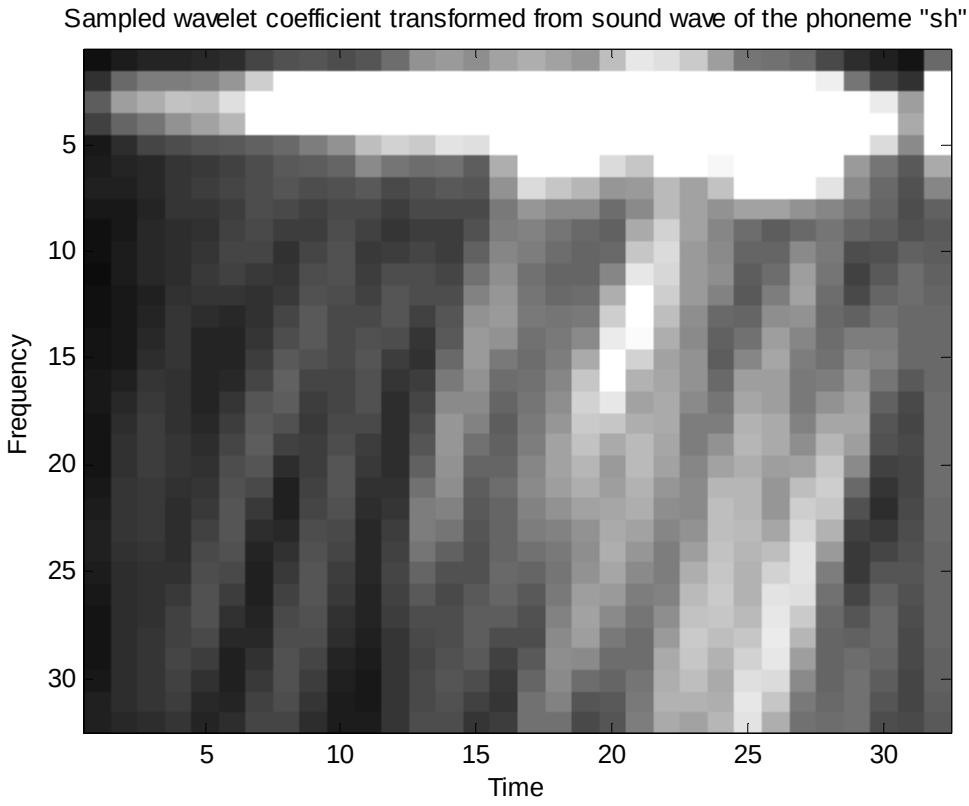


Figure 3. The image of sampled wavelet coefficient transformed from filtered sound wave of the phoneme “sh”.

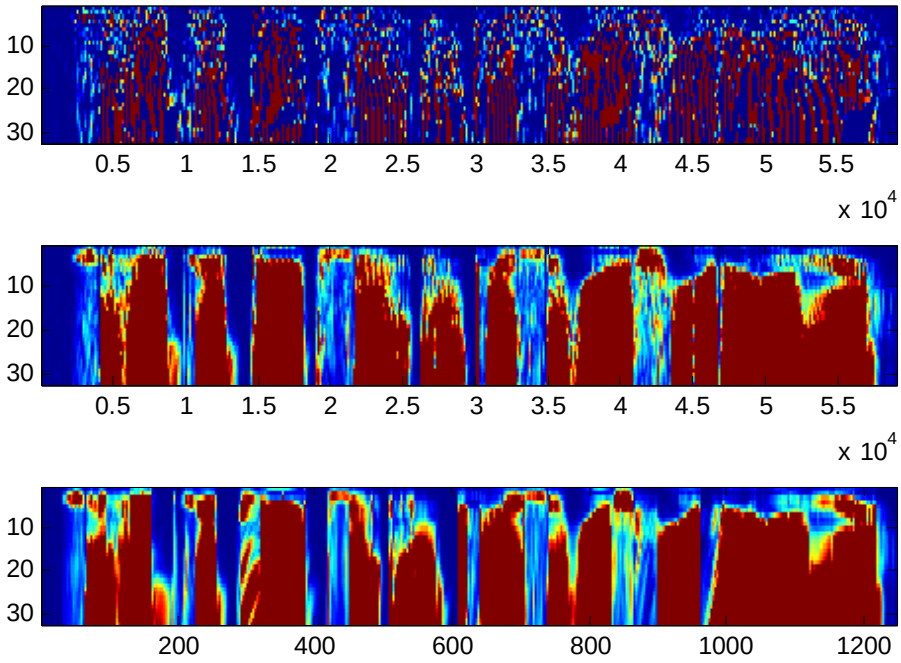


Figure 4. The image of sampled wavelet coefficient transformed from sound wave of a whole sentence.

Figure 4 shows the preprocessed sound wave of a whole sentence “She had your dark suit in greasy wash water all year”. The blue areas represent silence and the red areas represent the actual voiced phonemes. This figure also proves again that, the information of the sound wave passing through low-pass filter and digital sampler still contains significant contents for future analysis.

Since the input of neural network is a vector of the digitalized numbers, each sentence is translated into a 1x1024 vector from the 32x32 matrix. Therefore, a large matrix with a size of 1024 by (the number of sentences) composed by these translated vectors is

formulated for each training and testing set, and these matrices are saved into files to be ready for further use.

2.2 Multi-Layer Feed-Forward Neural Network

A Feed-Forward Neural Network, also known as multi-layer perceptron, is widely used in many practical applications. [Neu 2008] It is an interconnection of single layer perceptrons in which data and calculations flow in one direction from the inputs to outputs. [Num 2007]

The simplest network of this kind contains only one single layer of inputs and single layer of outputs. In this thesis, a 3-layer feed-forward neural network is used to classify the phonemes from the preprocessed sound wave. The 3-layers are the input layer which contains 1024 input neurons, the hidden layer with various hidden neurons, (where the number of hidden neurons are to be determined in the training process and need to be chosen for efficient processing), and the output layer containing 42 neurons which represent 42 different phonemes to classify.

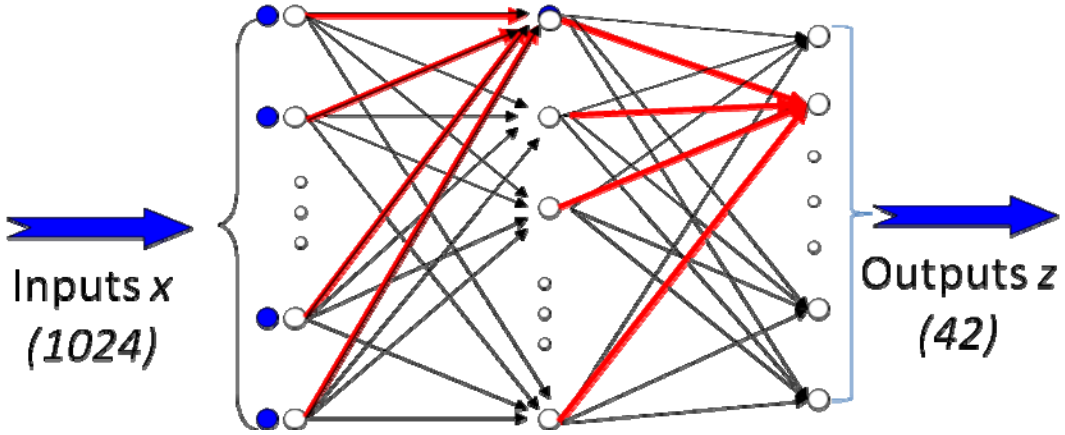


Figure 5. The structure of the multi-layer feed-forward neural network.

The input neurons fire at a certain excitation level with a strength corresponding to the preprocessed sound wave data. The information from the input neurons is summed with various weights to excite the hidden neurons in the hidden layer, and then each hidden neuron uses a non-linear activation function, also called the neuron output function, [Neu 2008] described by

$$\sigma(y) = \frac{1 - e^{-2y}}{1 + e^{-2y}}, \qquad (2.2)$$

which is a hyperbolic tangent sigmoid transfer function, where

$$y = \sum_{j=0}^{n} w_j x_j + b_j, \qquad (2.3)$$

and the transfer function of the hidden neuron in this thesis is described by

$$\Phi = \sigma\left(\sum_{j=0}^{n} w_j x_j + b_j\right), \qquad (2.4)$$

where w_j represents the weight of the connection between the *jth* input and this hidden neuron, b_j denotes the bias of the *jth* connection, and x_j is the *jth* input. Finally, the output neurons are obtained by summing the excitation strength of all the hidden neurons with corresponding weights. So the formula to obtain one output neuron z in a three layer neural network is:

$$z = \sum_{i=0}^{m} w_{2i}\sigma\left(\sum_{j=0}^{n} w_{1j}x_j + b_{1j}\right) + b_{2i} = \sum_{i=0}^{m} w_{2i}\Phi + b_{2i} \qquad (2.5)$$

We can also write this formula as follows:

$$Z = \Phi \cdot W + B \qquad (2.6)$$

After the excitation levels of the output neurons are calculated, a Winner-Take-All (WTA) method is used to find the strongest excitation neuron, which represents the recognition result.

In neural network systems, the connections between two layers are the core of the network structure, and the training process of the neural network is the process of obtaining the weights of these connections. Assuming that the bias b is negligible, then equation 2.4 can be written as:

$$Z = \Phi \cdot W = f(\phi) \quad (2.7)$$

From equation 2.5, the straightforward method to obtain the weights is to invert the input matrix Λ and then multiply it by the output vector:

$$W = \Lambda^{-1} \cdot Z \qquad (2.8)$$

where,

$$W = \begin{Bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & \ddots & & w_{2m} \\ \vdots & & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{Bmatrix} \qquad (2.9)$$

represents the connections between n inputs and m outputs,

$Z = \begin{Bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{Bmatrix}$ for m outputs, and $\Lambda = \begin{Bmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_n \end{Bmatrix}$ for n inputs.

There are several different algorithms for Multilayer Perceptron Learning, and in this thesis the back-propagation training algorithm is used. This algorithm contains two phases: the forward pass phase and the backward pass phase. In the forward pass phase, the input signals are propagated forward through the network, and then in the second phase, the errors are propagated backwards through the network [Erd 05]. The basic idea of this algorithm is to find out a way to modify the weights in order to reduce the errors, where the error function is:

$$E(t) = \frac{1}{k}\sum_{k=1}(d_k(t) - y_k(t))^2 , \quad (2.10)$$

and *d(t)* is the desired output value at time *t*, and *y(t)* is the current output value at time *t*. To find the minimum *E*, we can use gradient-based weights adjustment:

$$\Delta w_{ij} = w_{ij}(t+1) - w_{ij}(t) \propto -\frac{\partial E(t)}{\partial w_{ij}(t)}, \quad (2.11)$$

the problem is that this could go to local minima instead of the global minimum point, and it also could cost very long time to process due to the large number of learning steps

and the number of iterations needed. There is another efficient multi-layer perceptron learning method which is used here to obtain the minimum sum of squared error, hence to adjust the weights. It is known as the Least-Squares method for NN.

2.3 Least-Squares Method for Weight Adjustment

2.3.1 Introduction

Least-Squares method is used to obtain the minimum sum of squared errors for a system of linear equations. It can be used to solve both underdetermined and overdetermined systems. For an underdetermined system, Least-Squares method finds the solution with the minimum sum of weights using system regularization; and for an overdetermined system, the pseudo-inverse finds the solution with minimum sum of squared errors. In the neural system used in this thesis, the number of training data points is larger than the number of unknown weights in the NN and this produces an overdetermined linear system of equations. So the Least-Squares algorithm is applied in the optimization for weights of the connections between the layers. The Least-Squares Algorithm (LSA) is able to provide best fitting to go through all the points in this nonlinear system and the idea of the LSA is to adjust the function parameters to best fit a data set. The data set contains the input points and corresponding output points of each layer in the neural network, (ϕ_i, z_i), and $f(\phi)$ is the fitting curve. Suppose $f(\phi)$ has error e for each data point, i.e. $e_1 = z_1 - f(\phi_1)$, then according to the least squares algorithm, the best fitting curve should have the property of:[Lea 2008]

$$\mathrm{E} = e_1^2 + e_2^2 + \cdots + e_n^2 = \sum_{i=0}^{n} e_i^2 = \sum_{i=0}^{n} (z_i - f(\phi_i)) = \min \qquad (2.12)$$

2.3.2 Training algorithm

The training algorithm used in this thesis is the back-propagation algorithm that was first introduced by Yinyin Liu in [Liu 08]. First we randomly initialize the value of the weights and biases between the output layer and the second layer to obtain the signals on the second layer, then the weights and biases for the connections between the first layer and the second layer can be optimized from the random initial value using the input signals and the signals on the second layer. After these weights and biases are calculated, the signals on the second layer can be re-calculated from the optimized weights, biases and the input signals, finally the signals on third layer, (that is also the output layer in our case), can also be obtained using the optimized weights and biases between the second layer and this layer and the signals on the second layer. In general, if we have n layers, the weights and biases between each layer can be optimized and the signals on each layer can be conducted, in a feed-forward fashion (moving from lower to higher layers) until the output layer is reached. [Liu 08]

Suppose the input signals *x* and desired output signals *d* are provided from the training data in the database, W_{ab} and b_{ab} denote the weights and biases between the input layer a and the hidden layer b, and W_{bc} and b_{bc} denote the weights and biases between the hidden layer b and the output layer c accordingly. We also let y_1 and z_1 to represent the signals of

the hidden layer before and after the nonlinearity, and y_2 and z_2 to represent the signals of the output layer before and after the nonlinearity accordingly. Then the following procedure describes the least squares method of NN training used in this thesis to find all the optimized weights:

1.) Randomly initialize the weights and biases W_{ab}, b_{ab}, W_{bc} and b_{bc} in the network with the values in $[-\infty, \infty]$.

2.) Use the desired output signals *d* to calculate the signals *d'* obtained before the transfer function.

3.) Calculate the signal z_1 of the hidden layer after the non-linear transfer function using the least square method to satisfy:

$$W_{bc} \cdot z_1 + b_{bc} = d' \qquad (2.13)$$

The signal z_1 can be obtained by the pseudo-inverse of the augmented coefficient matrix:

$$\begin{Bmatrix} z_1 \\ 1 \end{Bmatrix} = \begin{bmatrix} W_{bc} & b_{bc} \end{bmatrix}^{-1} \cdot d' \qquad (2.14)$$

4.) Use the obtained signal z_1 to calculate the signal y_1 before the transfer function of the first layer, and z_1 needs to be linearly scaled into the range of the transfer function output.

5.) Now, we can optimize the weights and biases for the connections between the first layer and the hidden layer using calculated signal y_1 and input signals *X*:

$$\begin{bmatrix} W_{ab} & b_{ab} \end{bmatrix} = y_1 \cdot \begin{Bmatrix} x \\ 1 \end{Bmatrix}^{-1} \qquad (2.15)$$

6.) After W_{ab} and b_{ab} are obtained, the signal y_1 before the transfer function of the first layer can be calculated using these new values, and subsequently used to calculate the signal z_1 after the transfer function by the following formulas:

$$\begin{aligned} y_1 &= W_{ab} x + b_{ab} \\ z_1 &= f^1(y_1) \end{aligned} \qquad (2.16)$$

7.) As the new z_1 is obtained in 6.) and the signal d' found in 2.), we can optimize W_{bc} and b_{bc} using least-square algorithm through this formula:

$$\begin{bmatrix} W_{bc} & b_{bc} \end{bmatrix} = d' \cdot \begin{Bmatrix} z_1 \\ 1 \end{Bmatrix}^{-1} \qquad (2.17)$$

8.) The signals of the output layers y_2 and z_2 can now be calculated based on the current value of W_{bc} and b_{bc} using formulas:

$$\begin{aligned} y_2 &= W_{bc} z_1 + b_{bc} \\ z_2 &= f^2(y_2) \end{aligned} \qquad (2.18)$$

9.) The final step is to calculate the mean squared error (MSE) between the calculated output signals z_2 and the desired outputs d by using the following formula, and then normalize the MSE.

$$J = \frac{E\left|(d - z_2)^T (d - z_2)\right|}{E[d^T d]} \qquad (2.19)$$

The NN learning program iterates from the step 2 to step 9 for a number of trials until the MSE J reaches a certain minimum, hence, the final calculated output signals z_2 are most

likely equal to the desired outputs *d*. We can save the final weights and biases which result in these output signals as the best results. In the above procedure, “the desired outputs are propagated back through layers, and the Least Squared Method is used to obtain the optimal weights and biases. It yields a single global optimum solution for all stages of the applied procedure.” [Liu 08] Thus, by using this method, the global optimized weights and biases can be found.

2.4 Model Simulation and Discussion

For neural network learning and testing, a proper database selection is very important. In this thesis, TIMIT acoustic-phonetic continuous speech corpus was selected for training and testing. It contains broadband recordings of 630 speakers of eight major dialects of American English, and each reading ten phonetically rich sentences. Since the goal of this thesis is to recognize the phonemes not sentences, separating the phonemes from the training sentence samples is very important and is the most difficult part. There is no existing technology that can automatically separate either phoneme or words accurately enough from a spoken sentence, but the TIMIT database already provided the endpoints of each phoneme and each word in the sentences to make our task simpler.

By using the training algorithm mentioned above, the network can be constructed with the proper weight connections and biases which can result in optimal outputs. Next we send the testing samples into this network to obtain the recognition results.

2.4.1 Model Simulation

Training and testing samples:

80 sentences (total 2957 phonemes) from the training sets of the database

40 sentences (total 1497 phonemes) from the testing sets of the database

Neural Network:

Input: 1024 input neurons (size of the vector), which represent the preprocessed sound wave information of one phoneme.

Hidden layer: 1 layer contains 50—70 hidden neurons

Output: 42 output neurons which represent 42 different phoneme classes

Network structure: Weighted Least-Squares Multilayer Perceptron (3 iterations and 2 Monte-Carlo runs)

Training and testing results:

Table 1:

Training and testing results for phoneme recognition

# of Hidden Neurons	50	55	60	65	70
J_train (mse)	0.4883	0.4712	0.4691	0.4516	0.4378
J_test (mse)	1.2133	1.2183	1.2390	1.1846	1.2662
Misclassification (training)	342/2957	291/2957	273/2957	283/2957	212/2957
Misclassification (testing)	1050/1497	1023/1497	1014/1497	1027/1497	1030/1497
Testing Error	70%	68%	67%	69%	69%

From Table 1, we can conclude that more hidden neurons do not necessarily improve the classification results. The overfitting problem also occurs in this project.

2.5 Conclusion

Due to very high time and frequency variances between different people's speech, and even more variances between different dialects, the individual phoneme recognition performance is not good in the above table, although the method used can approach the optimal recognition results in theory. However, since only certain phonemes' combination in a specific sequence can compose a word, by applying a sequence learning method to phoneme recognition, the system can predict the next phoneme and correct the previous phoneme in the same word, hence, the performance of the recognition should be

improved. In Chapter 3, a sequence learning method will be introduced and a word level simulation will also be introduced after adding a sequence learning block on top of the phoneme recognition neural network for word recognition. In Chapter 4, I will apply this sequence learning method to the phoneme recognition system, and explain how to improve the phoneme recognition rate by adding the feedback signals from the sequence learning blocks.

3. LONG TERM MEMORY AND SEQUENCE LEARNING ALGORITHMS

3.1 Long Term Memory Block

3.1.1 Introduction to Long Term Memory

Research in our group focuses on Embodied Intelligence, so the simulation network structures and sequence learning blocks is modeled similarly to the human brain. As we know, humans have two major types of memory, Short-Term Memory and Long-Term Memory. In Wikipedia, Short-Term Memory is defined as the memory “to hold a small amount of information for about 30 second”[Wikis]; and Long Term Memory is defined as “the memory that can last as little as a few days or as long as decades”[Wikil]. Over the past decade, many scientists have studied these two types of memory trying to find a better understanding of human memory. It is believed that these two types of memory have different structure and functionalities. In this thesis the long-term memory structure is used to store the sequence information.

The basic idea of using the long-term memory (LTM) instead of short-term memory (STM) is to save the trained sequence information as long as possible, since in reality the system should be only trained once for as many different combinations of sequences as possible, then it will be used to predict the next testing sequence after all the training is done. The LTM is composed by a number of memory blocks, and each block saves the

letter/phoneme combination information of one dedicated sequence, while waiting to recognize the represented character.

3.1.2 The Structure and Operation of Long-Term Memory

The structure of the Long-Term Memory discussed in this thesis is based on the architecture of LTM introduced by J. Starzyk and H. He in [Sta 2007]. In that paper, the LTM structure is used to recognize text. But in this thesis, the technology was adapted to recognize spoken words, and provide feedback to correct phoneme recognition errors.

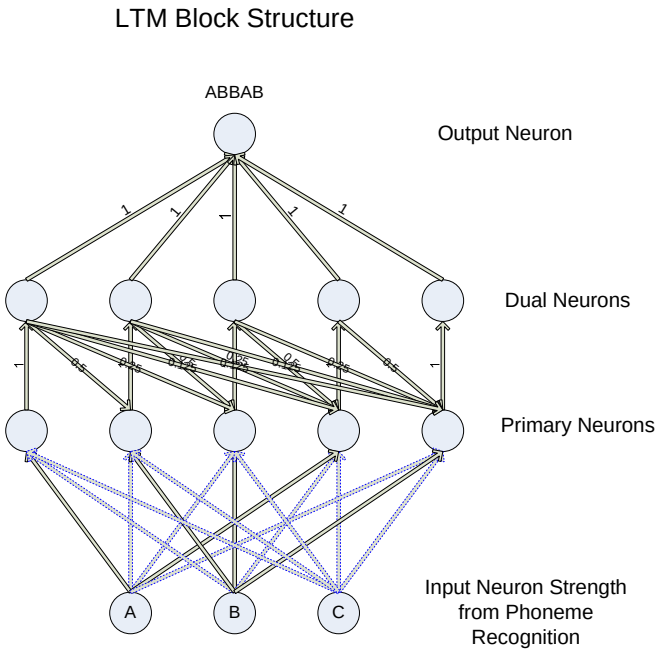


Figure 6. The basic structure of the memory block in LTM introduced in [Sta 2007].

Figure 6 shows a sample structure of the memory cell in the sequence learning block described in [Sta 2007]. In this thesis, the memory block structure is configured for speech recognition, i.e. instead of 26 inputs representing 26 letters, the inputs of each block $I = i_0, i_1, i_2 ... i_{41}$ are a series of neurons that represent all 42 phoneme recognition categories $Z = z_0, z_1, z_2 ... z_{41}$ from the neural network output. Since, in the training mode the number of elements in the sequence is known, the number of primary neurons in LTM equals to the number of elements. Each primary neuron connects to all the inputs with different weights (the excited inputs have weights of 1, and others are set to 0). Since the input sequence may be discontinuous or out of order due to the noise, dual neurons are used to save the excitation strength, and to help excite other elements in this saved sequence. Each dual neuron connects to its corresponding primary neurons with a weight of 1 and adjacent primary neurons with reduced weights defined by a weight reduction coefficient *q*, resulting in internal weights $W_{\text{int}\,ernal}$ (a more distant primary neuron receives a weaker feedback signal). The final output of each LTM cell is equal to the sum of the activations of all dual neurons. As a sequence is being trained, the excitation level of primary neurons is obtained by the input signal strength multiplied by the input weights, added with the feedback from previous dual neurons. And then the dual neurons simply copy the excitation strength of the corresponding primary neurons. The LTM response strength for a current element of the saved sequence is the sum of the excitation strength of all the dual neurons computed at the current time.

After the training process is finished, each LTM saves a strength vector for the dedicated sequence of elements. These values will be used as a reference to compare with the LTM response of testing samples.

In the testing mode, the same LTM structure is used to obtain the LTM response strength of the input sound, and compare (divide) it to the saved strength. The LTM response strengths for the currently playing input element are normalized between 0 and 1. If both testing and saved strength match, the LTM response strength is 1, otherwise it is less than 1 but always a non-negative number. Since each testing sequence needs to be compared to all the saved LTM's, the testing sequence may be shorter or longer than the saved sequence. Thus, in case of a shorter sequence, because the last element of the playing sequence is set to the end-of-sequence symbol, the final LTM response strength is reduced. For a longer sequence, the final signal is normalized using the logarithm of the length ratio. In order to keep the output signal from growing rapidly with the sequence length, the following equation is used to predict the LTM response for *ith* element of the saved sequence:

$$S_i = 1 + q \cdot r \cdot (\frac{2qr^{i-1} - 1}{2qr - 1}), \qquad (3.1)$$

where q is a weight reduction coefficient and r is equal to 1/decay factor.

After all the relative LTM response strengths of the playing elements have been obtained, a Winner-Take-All (WTA) method is used to choose the best fit LTM with the strongest

relative respond strength. The sequence stored in this winning LTM is the recognition result.

3.2 The Training Algorithm and Testing Process of Sequence Learning Block

In the above section, we have discussed a general structure of the Long-Term Memory cell used in sequence recognition. A detailed training algorithm and simulation process for sequence learning are introduced in this section.

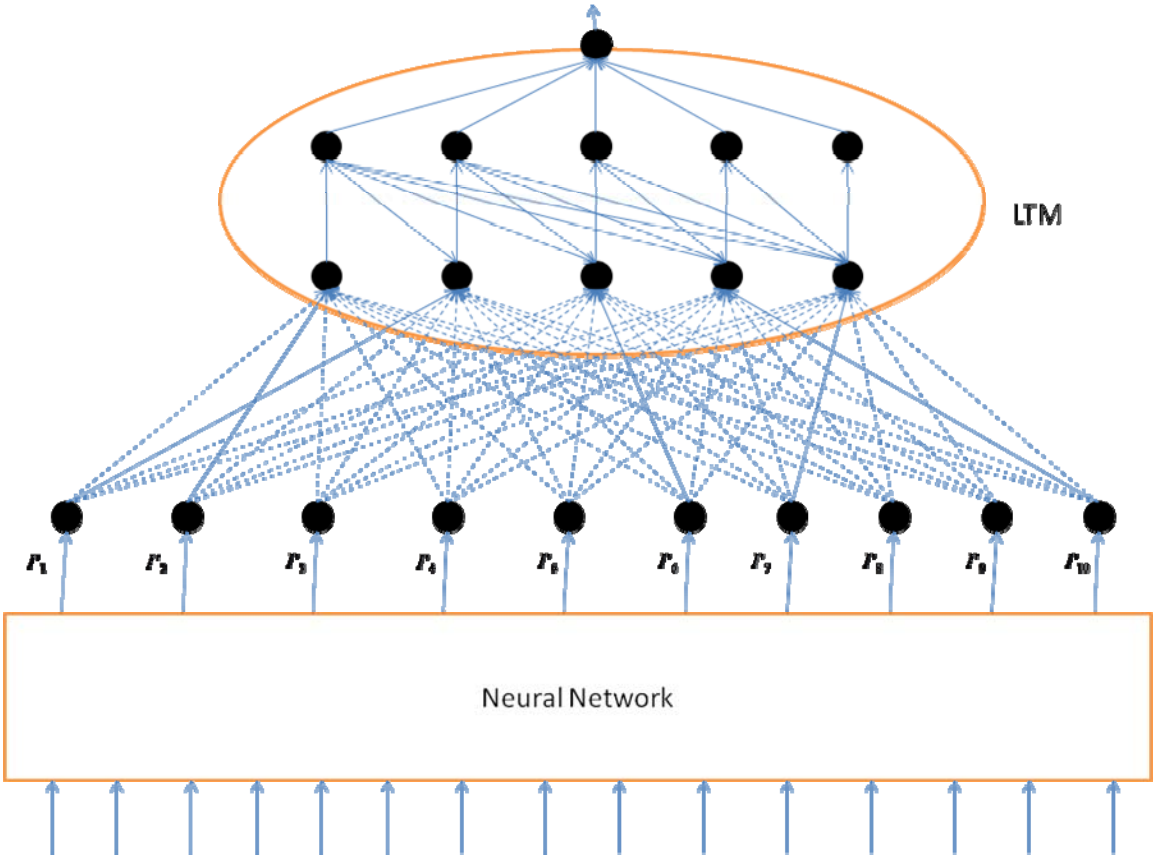


Figure 7. The detailed external connections between LTM and phoneme recognition NN and internal connection of LTM

TRAINING ALGORITHM

1. Set all primary neurons (PN) and dual neurons (DN) to zero. In order to be saved in one matrix for all LTM cells, we make the number of PNs and DNs the same in all the cells, so in this case, the number is set to 18 which is greater than the number of phonemes in the longest word possible.
2. Since in the training mode the desired activation output neuron from phoneme recognition is known, the weights for the connections between PN and the LTM cell inputs *I* can be obtained based on the desired outputs. E.g. if the first phoneme in the word activates the third output neuron, then the connection between the first PN in the current training LTM cell and the third input neuron has a weight of 1 and all other connections between the fist PN and the rest of input neurons have weights of zero, i.e.

$$W_{PN,1,3} = 1, W_{PN,1,1} = W_{PN,1,2} = W_{PN,1,4} = \cdots = W_{PN,1,42} = 0$$

3. There are two types of connections between the dual neurons and primary neurons, one is a feed-forward connection and another is a feedback connection. The PN signals are always passed feed-forward to DN's, and all the weights for feed-forward connections are set to 1; the feedback signals from DN's to PN's are used to pass the sequential information from previous phoneme to the next one, and the feedback weight between *nth* DN and *(n+m)th* PN is calculated by the following formula:

$$W_{\text{int}ernal,n,n+m} = \delta^m$$

where δ is the decay factor and $0 < \delta < 1$.

4. Repeat steps 2 and 3 until all the phonemes in the playing word have been input, then

the structure of one LTM cell has built up, (i.e. the weights for the connection between PNs and inputs, PNs and DNs, and DNs and outputs are all obtained.)

5. Repeat steps 1 through 4 until all the words in the training set have been trained, and then the sequence learning block is established with several LTM cells, where each cell saves the sequential information of one word.
6. As the internal structure of the sequence learning block builds up, the ideal/desired outputs I'_{NN} (only the activation neuron has a value of 1, the others are zeros) from the phoneme recognition NN are sent into the LTM cells as inputs for the sequence learning in the word level.
7. The *nth* primary neuron in each LTM cell first accepts the weighted inputs from the inputs of *nth* playing phoneme in the current training word and sums them up as the following formula:

$$PN_n = \sum_{i=1}^{42} I'_{NN,i} \cdot W_{PN,n,i} \qquad (3.2)$$

Next, the feedback signals from the previous dual neurons are also added into this *nth* PN. The strengths of these feedback signals for this PN are calculated by the following formula:

$$PN_{feedback,n} = \sum_{i=1}^{n-1} DN_i \cdot W_{\text{int}ernal,i,n} \qquad (3.3)$$

So the final strength of the *nth* PN is

$$PN_n = PN_{feedforward,n} + PN_{feedback,n} = \sum_{i=1}^{42} I'_{NN,i} \cdot W_{PN,n,i} + \sum_{i=1}^{n-1} DN_i \cdot W_{\text{int}ernal,i,n} \qquad (3.4)$$

Then the PN sends its strength to the corresponding dual neuron directly with weights of 1, i.e. $DN_n = PN_n$.

8. Now the strength of the output neuron for this *jth* LTM cell and nth playing phoneme can be calculated as:

$$B_{j,n} = \sum DN \qquad (3.5)$$

9. After all the phonemes in this word have been played, the output neuron has a vector of values which contain the LTM responses for each phoneme in this word. These responses are the desired reference values which will be compared with the responses calculated in step 13 to obtain the trained strength of LTM.
10. Since we want to save all the desired reference vectors in one matrix for simplicity, the maximum number of phonemes that can be saved in one LTM is set to a large number (18 in this case). In order to calculate the desired reference value which goes beyond the input phonemes in a word, we use the equation 3.1.
11. Repeat steps 6 through 10 for all training phoneme samples to obtain the desired reference values of all LTM cells in the sequence learning block.
12. Still in the training mode, we send the real-time output values from the neural network for the training samples into the sequence learning block, and repeat step 6 through step 10 for these data, to obtain the response B_{real} for each LTM. Then we divide these response values by the corresponding reference values $B_{desired}$, and these results are the final reference LTM response that will be compared by the testing results to see which LTM has the best match.

$$B_{\mathrm{Re}\,f} = B_{real} / B_{desired} \qquad (3.6)$$

where $0 \le B_{ref} \le 2$.

TESTING OPERATIONS

Now we can start to test the system. Use the real time NN output data of the testing set to send into sequence learning block, repeat step 6 through step 10 to obtain the responses B_{test}. Then divide these responses by all the final reference LTM response from training to obtain the relative responses. Finally, we use the Winner-Take-All method to find the best matching sequence based on the relative responses of all the LTM cells. The best match for the currently played test word sample is the sequence information saved in the LTM which has a relative response closest to 1.

3.3 Simulation Results of the Sequence Learning Block

A simulation is constructed as follows to show the performance of the sequence learning block. For simplicity, I use text converted inputs instead of phoneme recognition data since text input has less classification categories than phoneme, and both of them have similar characteristics.

Suppose that the word “perforation” was successfully saved in one LTM cell after training, the total energy of all input neurons is set to one (i.e. $E = \sum_n I_n^2 = 1$), the desired

activated input neuron $I_{21} = 1 - \alpha$ where α represents the noise level, and the other input neurons are all randomly generated with total energy of α (i.e. $E = \sum_{n=1}^{20} I_n^2 + \sum_{n=22}^{26} I_n^2 = 1$).

Table 2:

Results for the saved text "abba" with decay rate of 2

Input Text	Output Strength				
	1st	2nd	3rd	4th	5th
abba	1.0	1.0	1.0	1.0	
aabb	1.0	1.0	0.8654	0.8857	
bbaa	1.0	1.2	1.0750	1.0593	
baab	1.0	1.1	1.025	0.8854	
aba	1.0	1.0	0.9625		
abbab	1.0	1.0	1.0	1.0	0.9117
prus	0	0	0	0	

Table 3:

Results for the saved text "abab" with decay rate of 2

Input Text	Output Strength				
	1st	2nd	3rd	4th	5th
abba	1.0	1.0	1.0789	0.9669	
aabb	1.0	1.0476	1.0132	1.0331	
bbaa	1.0	1.0476	0.9474	0.9256	
baab	1.0	0.9524	1.0	0.9587	
aba	1.0	1.0	1.0		
abbab	1.0	1.0	1.0789	0.9669	0.8815
prus	0	0	0	0	

From the above 2 tables, Table 2 and Table 3, we can see that when the playing sequence exactly matches the saved sequence, the LTM response is 1, and when the playing sequence is different than the saved one, the strength of LTM response deviates from 1.

We use the following plots to show the LTM responses for the inputs with different levels of noise α in the input data which satisfies the equation of:

$$\sum (I_{NN,excited} + \alpha \cdot I_{NN,non-excited}) = 1$$

For playing the exactly same word “perforation” as trained sequence:

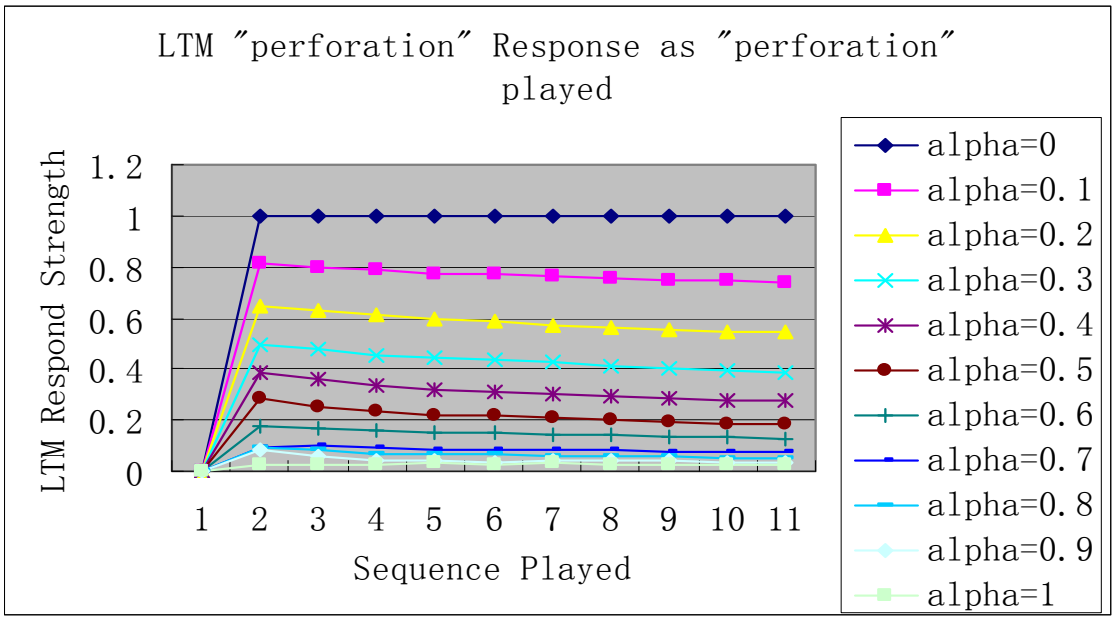


Figure 8. The plot of LTM cell “perforation” response for playing sequence “perforation”

For testing word “performance”:

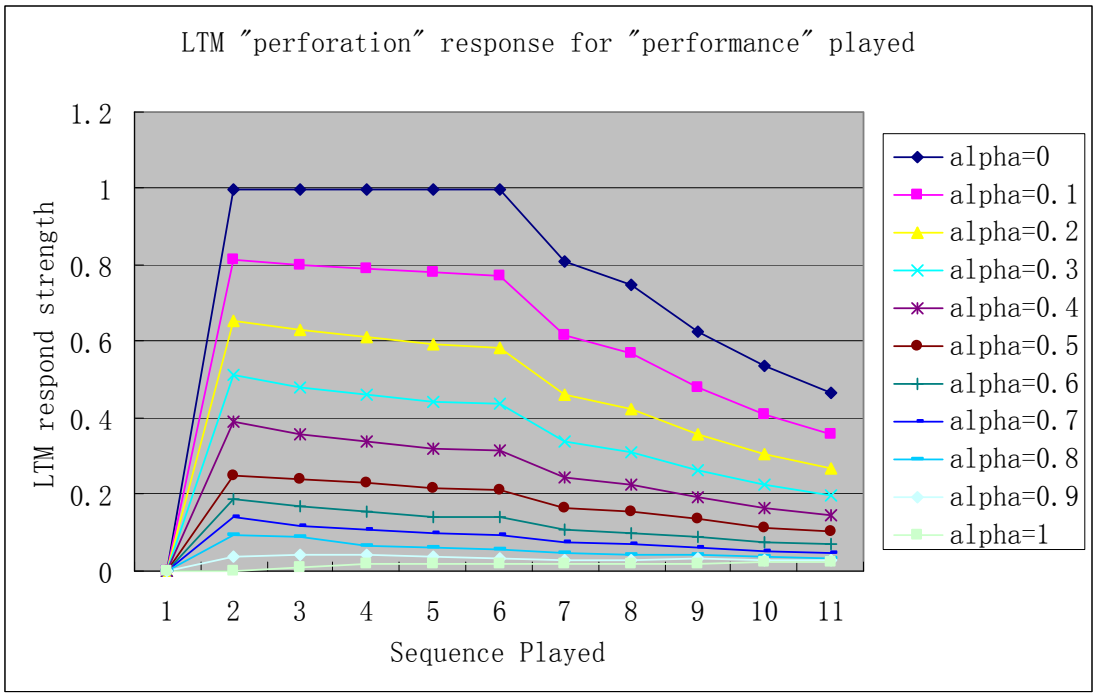


Figure 9. The plot of LTM cell “perforation” response for playing sequence “performance”

In an ideal situation, the noise level α is zero, the LTM response strength is 1 for the matched sequence (from letter 1 to letter 6), and is being reduced as different letters are played. As the noise level increases, the strength of the activated neuron decreases, so the LTM response strength reduces accordingly. From the results shown in Figure 8, we can see that even with the noise level of 60% the LTM cell still has a sharp decrease in activation value when the non-matching letters are played.

The following figure shows how well the LTM cell separates the matched and non-matched sequences. The plot shows the LTM output signal strength of word recognition by the entire sequence learning block with different noise levels:

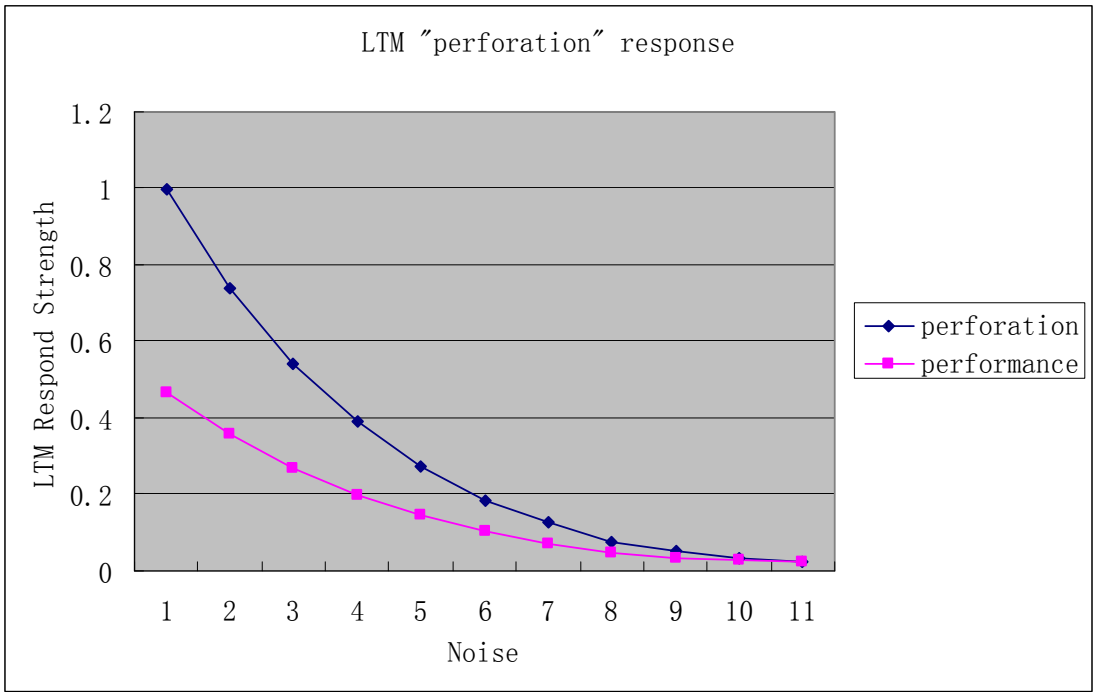


Figure 10. The figure showing the difference between the matched sequence and non-matched sequence ("perforation" vs. "performance").

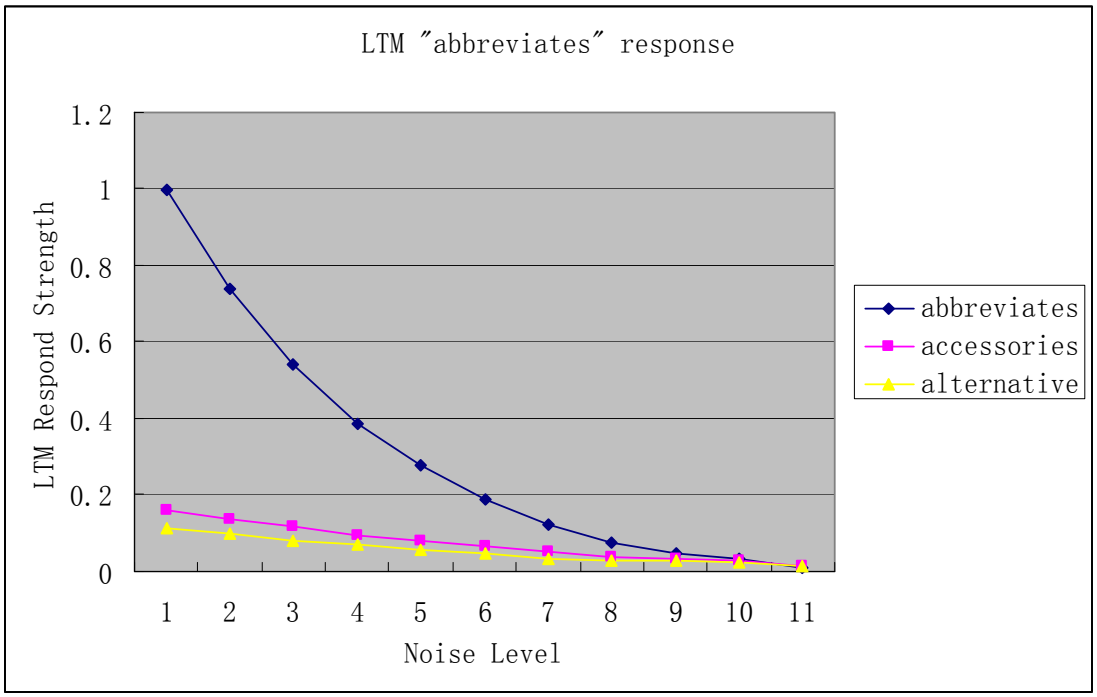


Figure 11. The figure showing the difference between the matched sequence and non-matched sequence ("abbreviation" vs. "accessories" and "alternative").

From Figure 9 and Figure 10, we can see explicitly that the sequence learning method can significantly recognize the matched sequence when noise level is less than 80%, i.e. when the recognition rate in the previous stage is significantly high, then the sequence learning method is also able to improve the recognition rate significantly, but as the recognition rate of the previous stage is getting lower, this method can provide less improvement to the whole recognition system.

From the results shown above, we can predict that by adding the sequence learning method to the recognition system the recognition rate can be improved to a certain level

depending on the recognition rate obtained from last stage. Adding another sequence learning block on top of the current system for the sentence level recognition, we can expect even better recognition rate.

4. THE PHONEME RECOGNITION CORRECTION METHOD USING FEEDBACK SIGNALS

4.1 Introduction

In this thesis, the phoneme recognition system introduced in Chapter 2 does not provide a good recognition rate for our simulation due to the variance in the speakers with different dialects, but in reality the recognition system needs to be adaptive to all types of speakers regardless of race and dialect. So a correction method that can improve the phoneme recognition rate needs to be considered and developed.

In Chapter 3, we showed that the sequence learning method can recognize the playing sequence with a certain level of noise, and that by adding the sequence learning block on top of the phoneme recognition neural network we can expect that the recognition rate at the higher level should have a significant improvement. Can this block provide help to the phoneme recognition block to improve its recognition rate? The answer is affirmative. However, the previous sequence learning block needs to be modified so that it can provide feedback signals to the outputs of the phoneme recognition network in order to predict the next playing phoneme, and to improve the overall phoneme recognition rate.

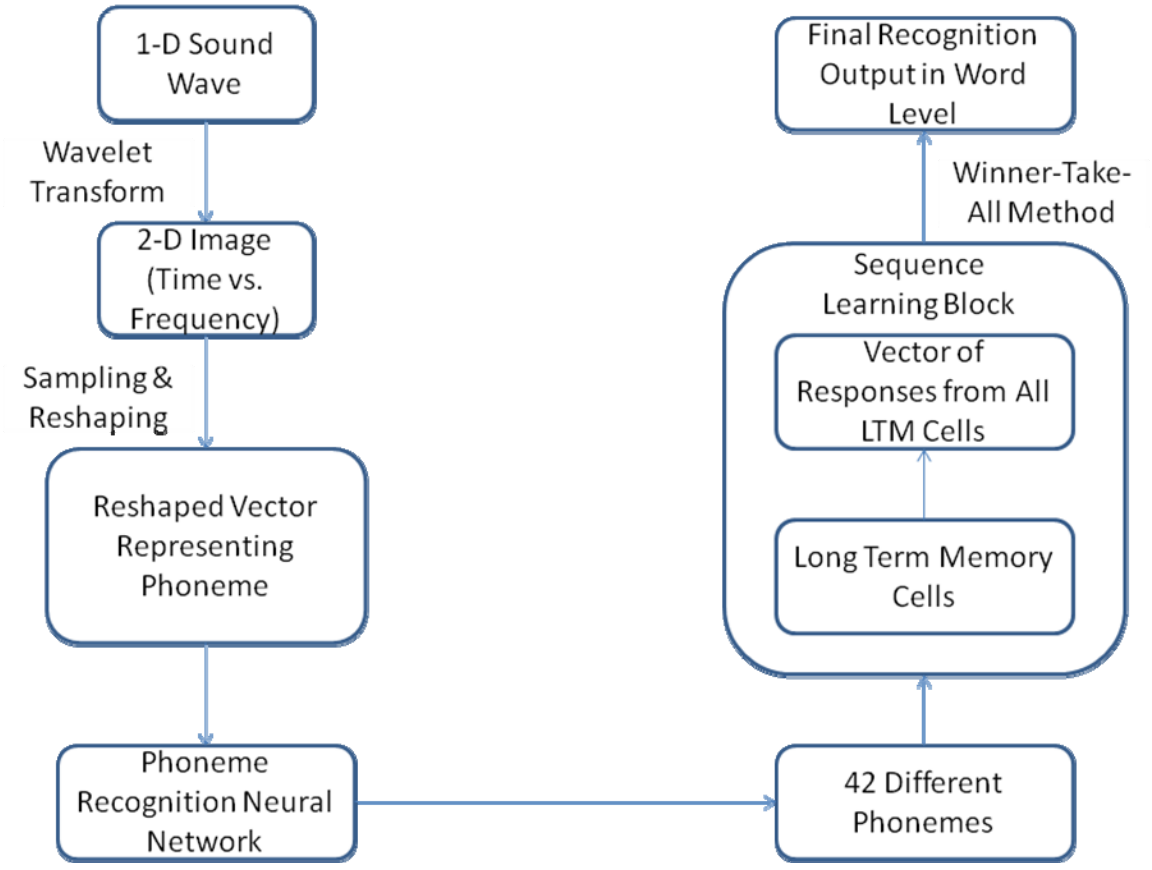


Figure 12. The flow chart of proposed word recognition system with phoneme recognition network and sequence learning block introduced in Chapter 3.

4.2 The Modified Sequence Learning Block with Feedback Signals

In the original sequence learning model described in the previous chapter, for each playing element, there is only one output response value from each LTM cell which is the summation of the strength of all the dual neurons in this cell. And the final recognition decision is made according to the largest response strength from all the LTM cells using the Winner-Take-All method. This original model only works for recognizing a sequence or making prediction to the next level. So in order to predict an individual character or

phoneme that is at a lower level than the current one by using this sequence information, some modifications are needed.

The prediction of the next character or phoneme in sequence learning always requires the information of the previously played characters because each sequence of words is unique. Since in the structure of the LTM introduced in the last chapter the LTM responses of the individual playing phonemes in one word is saved into a vector, we can create feedback signals from the LTM to all the input neurons, and these feedback signals provide the sequence information to add to the next playing phoneme for prediction.

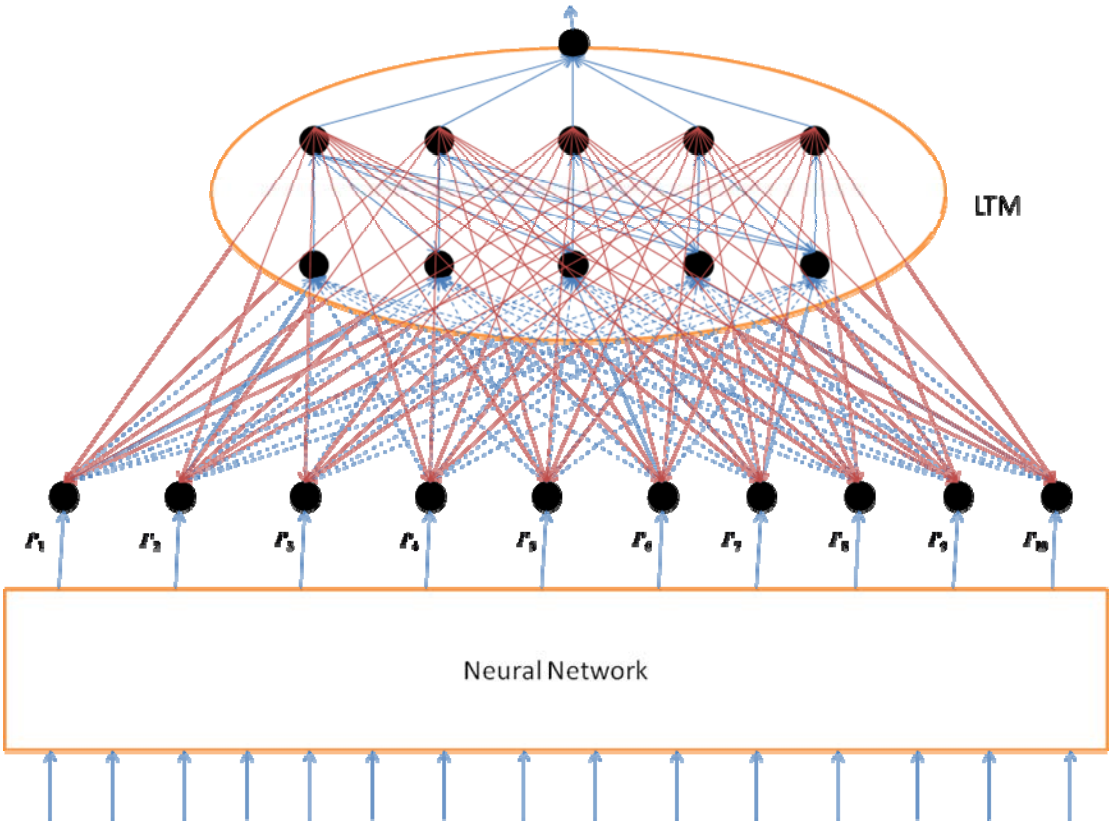


Figure 13. The detailed external connections between LTM and phoneme recognition NN and internal connection of LTM with feedback connections.

4.2.1 The Phoneme Recognition Correction Algorithm

The first step of this algorithm is to obtain the desired LTM responses in every LTM cell for all the training samples by sending the desired neural network outputs I'_{NN} (i.e. value 1 for desired firing neuron and value 0 for all other neurons) into the sequence learning block introduced in last chapter. All the LTM responses are saved into a vector S_{ref} which will be used later to get relative response of training samples in the next step.

Next, the real-time normalized NN outputs I_{NN} (i.e. all neurons fire with a non-negative strength with values between 0 and 1) from the same training samples as in the previous step are sent into the sequence learning block. Then by going through each LTM, the activation strengths of the LTM responses to the training elements can be obtained. Dividing these activation strengths by the desired response strengths will give us the final reference LTM response strengths $S': S'_{n,m}$.

From the last steps, we have obtained the LTM response strengths for the real-time inputs, and since the purpose of the feedback is to reduce the error between the desired inputs and real-time inputs, the real-time inputs should be as close to the desired inputs as possible by adding the feedback signals. So the weights of feedback connections can be obtained using the errors between the real-time inputs and desired inputs and the LTM response strengths. As the weights get calculated, the training stage is finished.

The following example is used to clearly describe how to find the weights of feedback connections in training. In this example a text sequence instead of phoneme sequence is used for simplicity, since they have the same sequence information.

Example

Suppose matrix A contains the location information and response strengths of the training sequences, and current training sequence is “perforation”:

We start with letter “p”. Since this is the first letter and the letter following is “e” which is the fifth letter in the English set of 26 letters, the response strength is placed in row 5; because this is the response for the first letter from the first LTM, the strength is placed in column 1 of the matrix A:

$A(5,1) = S_{1,1} \rightarrow A(26\times(1-1)+5,(1-1)\times 18+1) = S_{1,1}$, where 26 is the number of English letter categories and 18 is the maximum length of the sequence saved in the LTM cell.

For the same letter, the response strength from the second LTM cell is placed in row 5 and column 19:

$A(5,19) = S_{2,1} \rightarrow A(26\times(1-1)+5,(2-1)\times 18+1) = S_{2,1}$,

So we can conclude that the response from nth LTM cell is placed in row 5 and column $(n-1)\times 18+1$:

$A(26\times(1-1)+5,(n-1)\times 18+1) = S_{n,1}$.

The second training letter in this word is “e”. Since the next word after “e” is the eighteenth letter in English letter, “r”, use the same method as above, the response from the first LTM cell is placed in row 70 and column 2:

$A(70,2) = S_{1,2} \rightarrow A(26\times(2-1)+18,(1-1)\times 18+2) = S_{1,2}$,

and the equation for the second LTM cell response is:

$A(26\times(2-1)+18,(2-1)\times 18+2) = S_{2,2}$,

So the equation for *nth* LTM cell is:

$A(26\times(2-1)+18,(n-1)\times 18+2) = S_{n,2}$.

Thus, we can conclude a general formula for the nth LTM cell response strength for the *mth* training letter as:

$$A(N_{category}\times(m-1)+K_{desired,m+1},(n-1)\times L_{LTM}+m) = S_{n,m} \qquad (4.1),$$

where $N_{category}$ denotes the number of recognition categories, $K_{desired}$ represents the desired input neurons, and L_{LTM} is the maximum length of the sequence saved in all LTM cells.

Next we calculate the difference between the desired input of the sequence learning block from the neural network I'_{NN} and the real-time NN outputs I_{NN} for all training samples $E = I'_{NN} - I_{NN}$. The obtained results are the errors to be reduced in NN learning. Then a vector B is formulated for the feedback structure by reshaping the error matrix *E*

to a vector B with size of 1 by $N_{Category} \times N_{train}$, where N_{train} denotes the number of training letters/phonemes. As soon as the matrix A and vector B are obtained for the training set, the equation to calculate the weights $W_{feedback}$ of the feedback connections to NN outputs can be formulated as the following:

$$W_{feedback} = (B - I_{NN}) \times A^{-1}. \qquad (4.2)$$

Now the training stage of the sequence learning method with feedback signals is finished.

The testing process begins from here. In this process, the NN recognition outputs for the testing phoneme are sent into the sequence learning block individually. Every LTM cell has different response for the playing test phoneme, and this strength multiplied by the feedback weights to provide the values of the feedback signals. Each LTM cell has feedback connections to all the input neurons with different weights, so when the next NN outputs come in it will add these feedback signals to formulate the final input signals for the sequence learning block. Hence, this new input signals can also be seen as the corrected phoneme recognition results.

In test operation, unlike training, the phonemes are sent into the LTMs one by one in order for the system to predict the next playing phoneme. For simplicity, the word separation is assumed to be known. Then with these assumptions, the following test procedure is applied to obtain the test results:

1. The phoneme recognition neural network generates the input vector I'_{test} to the sequence learning block for the current playing phoneme from the testing set, which

contains different real values between 0 and 1. I'_{test} is sent into the LTM cells to obtain the response strength S'_{test}, which is a vector of 1 by $(L_{LTM} - 1) \times N_{LTM}$ to save the LTM response in all the LTM cells, where N_{LTM} denotes the number of LTM cells saved in the sequence learning block. Use the same formula 4.1 as in training to generate A_{test}, where

$$A_{test}(N_{category} \times (m-1) + K_{desired,m+1}, (n-1) \times (L_{LTM} - 1) + m) = S'_{test,m} \qquad (4.2)$$

2. Use the formula

$$B_{test,m+1} = I'_{test,m+1} + A_{test} \times W_{feedback} \qquad (4.3)$$

to calculate the next NN output vector $B_{test,m+1}$.

3. Substitute $B_{test,m+1}$ for I'_{test} in step 1, and repeat step 1 to step 2 to update S'_{test}, then A_{test} and $I'_{test,m+2}$, until all the phonemes in the testing word are played (i.e. reaches a long silence).
4. Reset all the internal neurons (PNs and DNs) of all LTM cells, and then repeat step 1 through step 3 for the new word

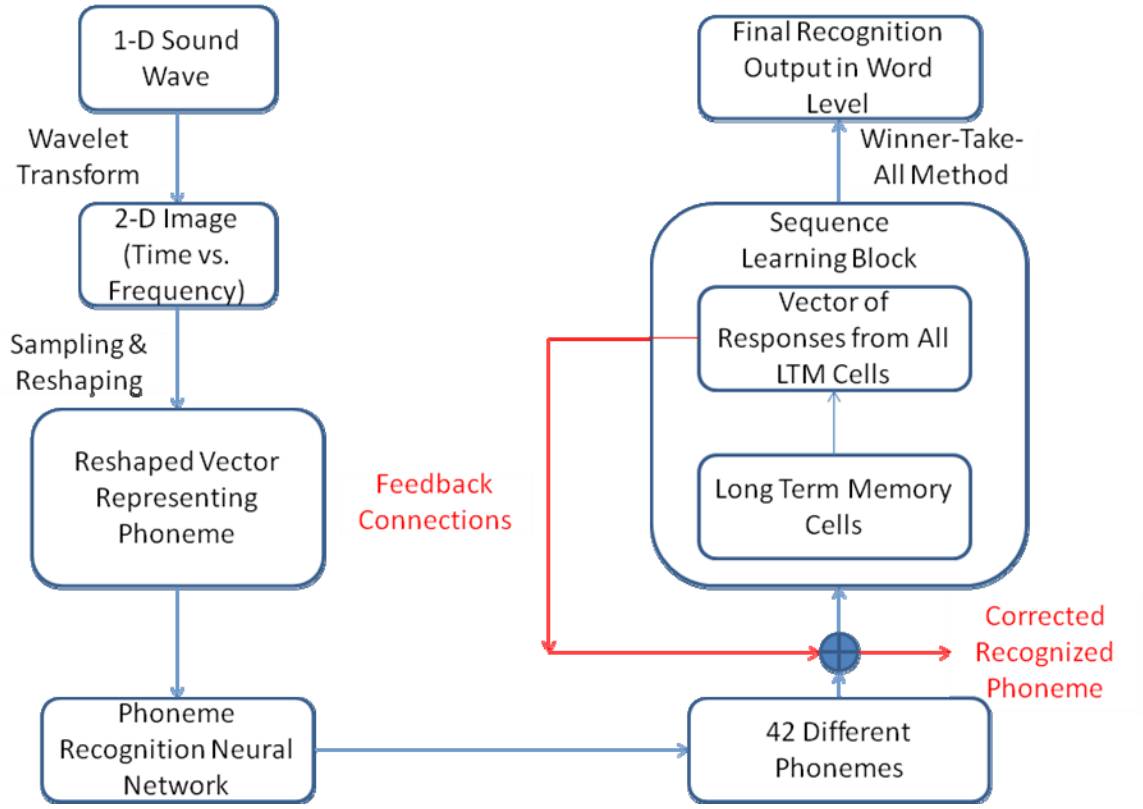


Figure 14. The flow chart of proposed phoneme recognition system with feedback correction method.

4.3 Simulation Results and Discussion

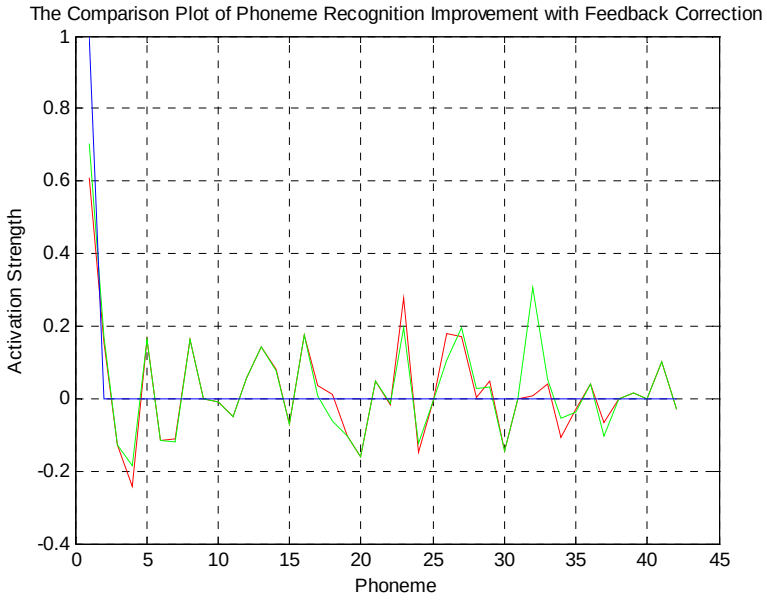
We sent the same phoneme recognition results obtained in Chapter 2 into the sequence learning block for simulation. The following results were obtained:

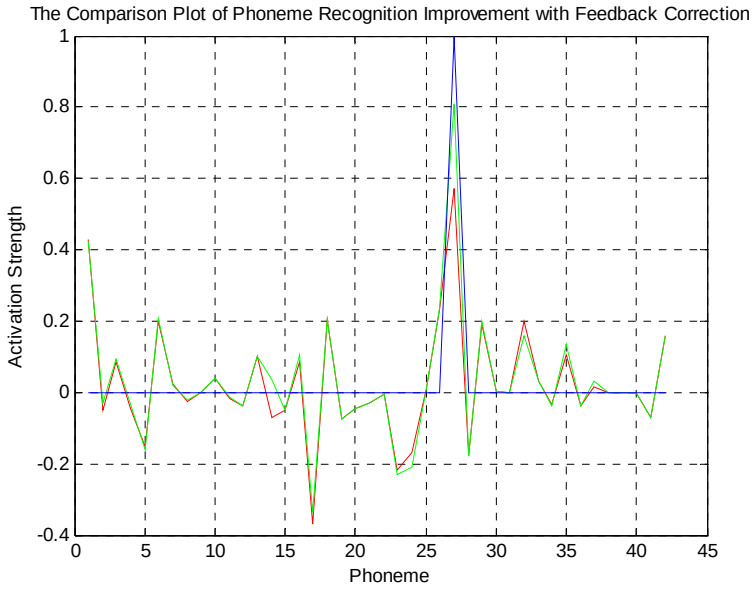
Table 4:

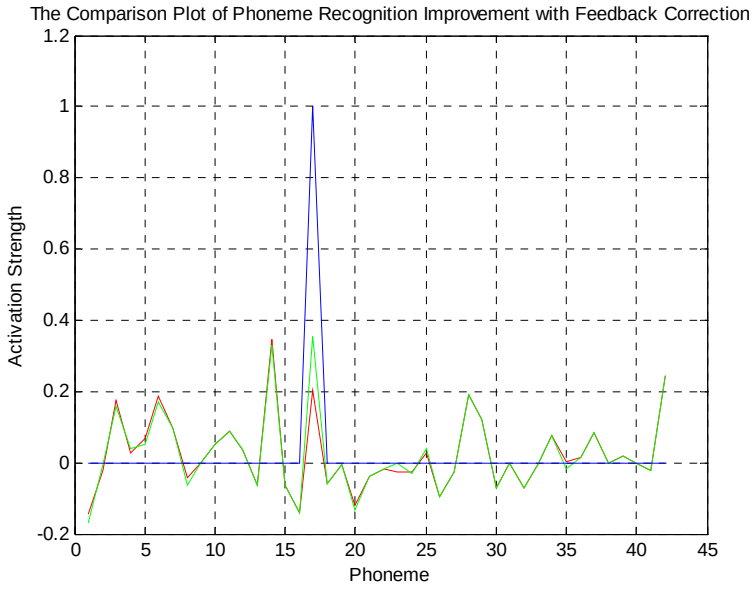
Comparison table of testing results for phoneme recognition with and w/o feedback correction involved

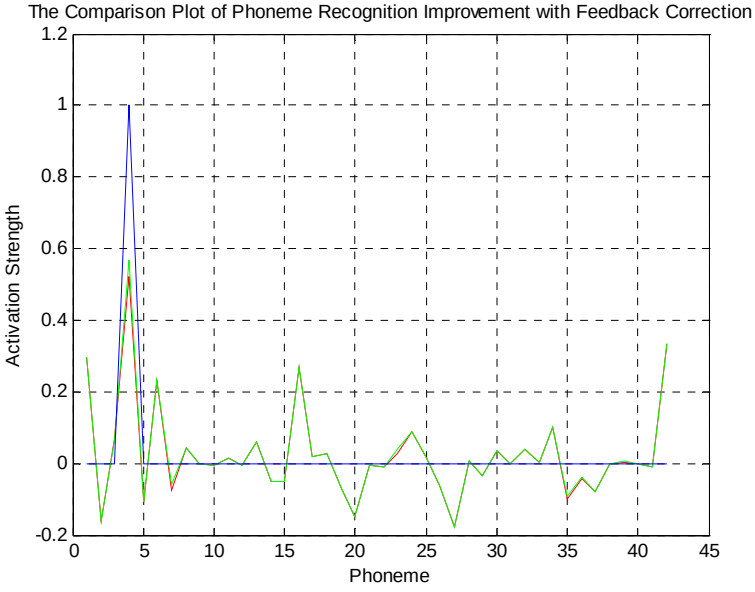
# of Hidden Neurons	50	55	60	65	70
Misclassification (Without Feedback)	1050/1497	1023/1497	1014/1497	1027/1497	1030/1497
Testing Error (Without Feedback)	70.14%	68.34%	67.74%	68.60%	68.80%
Misclassification (With Feedback)	1001/1497	972/1497	950/1497	976/1497	978/1497
Testing Error (With Feedback)	66.87%	64.93%	63.46%	65.20%	65.33%

The above Table 4 clearly shows that by adding feedback connections from the sequence learning block to the phoneme recognition block, the average phoneme recognition rate improves approximately by 5%. If we analyze each phoneme individually, we can see that most of the phoneme recognition results get improved by a certain level (i.e. the desired activate neuron gets promoted and other neurons get depressed) although the recognition results for many phonemes didn't improve. The following plots clearly show this result (Figure 15 continues from pages 61-63):


The Comparison Plot of Phoneme Recognition Improvement with Feedback Correction
Activation Strength
Phoneme
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
5
10
15
20
25
30
35
40
45


The Comparison Plot of Phoneme Recognition Improvement with Feedback Correction
Activation Strength
Phoneme
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
5
10
15
20
25
30
35
40
45


The Comparison Plot of Phoneme Recognition Improvement with Feedback Correction
Activation Strength
Phoneme
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15
20
25
30
35
40
45


The Comparison Plot of Phoneme Recognition Improvement with Feedback Correction
Activation Strength
Phoneme
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
0
5
10
15
20
25
30
35
40
45

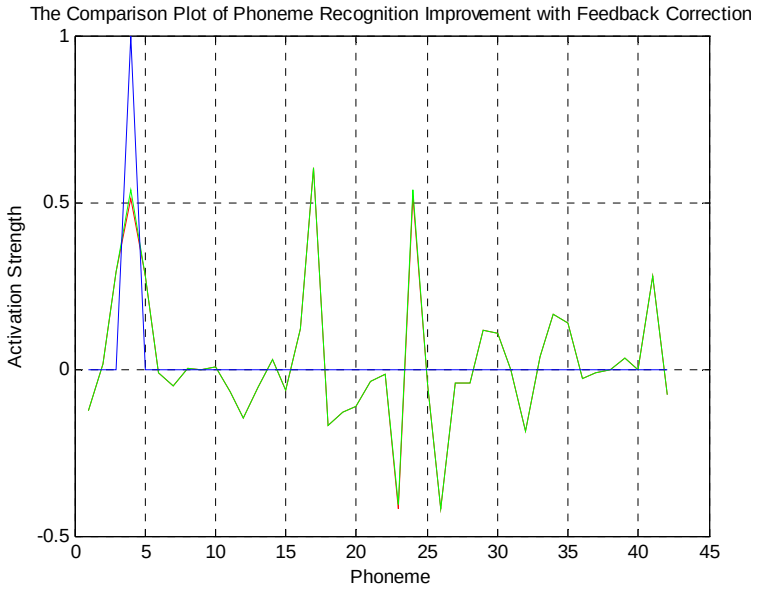


Figure 15. The comparison plots of phoneme recognition improvements with feedback connections: the blue line shows the desired results; the red line shows the results before adding sequence learning method; and the green line shows the final results with the feedback correction from the sequence learning block.

In Figure 15 above, the five plots display the comparison of the phoneme recognition results with and without the feedback correction for five different phonemes, and as we can see that all of the plots show different levels of corrections: they lower the activation strength of the inactive neurons and increase the strength of the active neuron. Plot #3 in Figure 15 shows a very good example of the corrections, which clearly shows the recognition results get corrected after feedback is involved.

4.4 Conclusion

Based on the simulation results obtained above, it can be proven that the sequence

information can provide some improvement to the phoneme recognition because of the sequence characteristics. The longer sequences provide more information and show more improvement because they are more unique. However, in reality, most common words are composed of less than 6 phonemes, so the sequence information provided is very limited. Therefore, by adding a sequence learning block to the phoneme recognition system, the recognition rate is improved, but with limited corrections.

5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

This thesis introduces an innovative phoneme recognition model that using a feed-forward neural network with a Least-Squared method for first level phoneme recognition, a sequence learning block with the structure of the Long Term Memory cells for word level recognition, and then combine these two parts with a feedback correction method to achieve an improvement of the phoneme recognition.

The first stage of this phoneme recognition model is a feed-forward neural network. To construct this network, we used the Least-Squared method to find the weighted connections between the hidden layer and input/output layers, which result in the best recognition rate. By using this method, the lowest number of the hidden neurons needed for the best match results are also found. At the output layer of the network, the Winner-Take-All method is used to find the strongest activation strength of the output neuron, and the phoneme represented by this neuron is the recognition result.

Since the performance obtained from the first stage is not sufficient to be close to other researchers' model [Yuk 1999] [Fal 1990], which can reach 62.2% and 66.2%, we introduced a sequence learning algorithm to be added on top of phoneme recognition neural network in order to improve the recognition rate.

In the sequence learning stage, a number of Long-Term Memory cells are used to compose the sequence learning block. These LTM cells have a similar structure as the LTM introduced in [Sta 2007] with some modifications to make these cells useable in speech recognition. By sending the outputs of the phoneme recognition neural network into this sequence learning block, word level recognition results can be obtained. The recognition rate for this higher level is better than for the lower level because the sequence information has been added. Hence, we can propose that the recognition rate should see more improvement at the sentence level because of more unique sequence information added from the word level.

Since my thesis focuses on phoneme recognition, I have also proven that the sequence learning method can help to improve phoneme recognition by sending feedback signals to the outputs of the neural network. In Chapter 4, the structural modifications from the model in Chapter 3 were described, the response strength of all the LTM's for any phoneme played needed to be tracked and send back a corresponding feedback signal to predict the next playing phoneme. The outputs of the neural network for the current playing phoneme plus the feedback signals from all of the LTM cells for the previous played phoneme generate the corrected recognition results for the playing phonemes.

5.2 Future Work

Although a certain level of phoneme recognition has been reached in this thesis, the

recognition rate is still not as good as the models in [Yuk 1999] and [Fal 1990], and far from commercial requirements. So additional future work is needed to further improve the recognition rate.

There are several ways to improve the recognition rate based on the model described in this thesis. In the sound preprocessing stage, instead of using the wavelet transform method, we can extract the features that represent the characteristics of the sound wave, and use a separate neural network for each feature in parallel. This way, the training time will be much less because of parallel processing, and the recognition rate will also be improved since all the extracted features will be unique to each phoneme.

A better algorithm to calculate the internal weights of the neural network also needs to be developed, since the current algorithm is not quite time efficient. The current processing time using least squared method increases rapidly when the number of training samples increases, so very limited training samples have been used in this thesis. This also adds to the poor testing performance. If a more efficient training algorithm is used, and the system saves more phoneme information and knowledge from training, the recognition rate should increase accordingly.

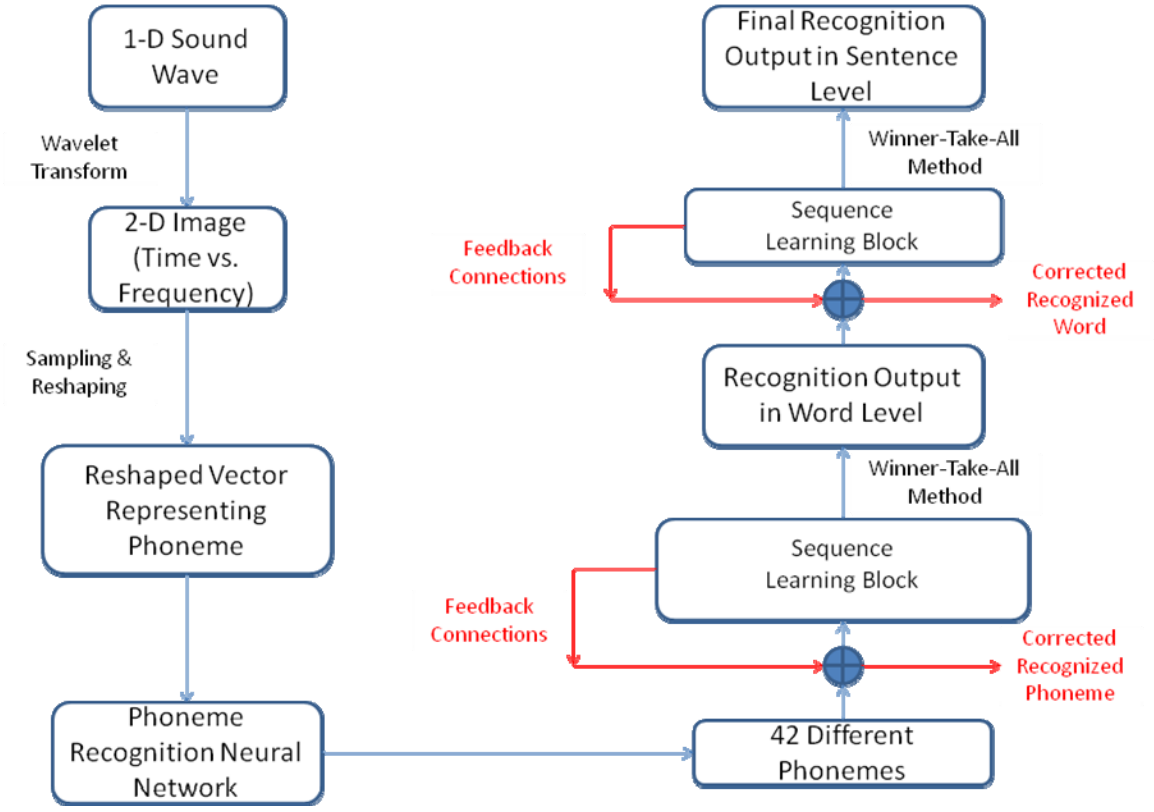


Figure 16. A general view of the proposed speech recognition system model.

Also, as shown in Figure 16, if we add another sequence learning block for sentences, and use feedback signals to correct the word level recognition, then use the corrected word level recognition results as the feedback signal to sent back to phoneme level, the phoneme recognition rate should be improved.

Another aspect of the future work for this thesis is to implement a hardware system to substitute the software simulation, so that it can be embedded into a large hardware "Brain" in the future.

REFERENCES

[Kom 1998] E. Komissarchik, V. Arlazarov, and D. Bogdanov, "Knowledge-based speech recognition system and methods having frame length computed based upon estimated pitch period of vocalic intervals" *U.S. Patent 5799276*, August 25, 1998.

[Wha 1988] National Center for Biotechnology Information, "What Are the Advantages and Disadvantages of the Different Types of Cochlear Implants?", *NIH Consens Statement Online*, May 2-4, 1998.

[Lon 1996] C. J. Long and S. Datta, "Wavelet Based Feature Extraction for Phoneme Recognition", *Proceedings of 4th International Conference of Spoken Lauguage Processing*, Philadelphia, PA, 1996.

[Jan 1996] L. Janer, J. Marti, C. Nadeu, E. Lleida-Solan006F, "Wavelet Transforms for Non-Uniform Speech Recognition Systems", *Proceedings of 4th International Conference of Spoken Lauguage Processing*, Philadelphia, PA, 1996.

[Dau 1992] Daubechies, I. *Ten Lectures on Wavelets*. SIAM, Philadelphia, PA. 1992.

[Woo 2001] S. C. Woo, C. P. Lim, R. Osman, "Development of a Speaker Recognition System using Wavelets and Artificial Neural Networks", *Proceedings of 2001 International Symposium*, 2001.

[Tan 1994] B. T. Tan, M. Fu, A. Spray, P. Dermody, " The use of wavelet transforms in phoneme recognition", *Proceedings of ICSLP 96*, 1996.

[Neu 2008] WolfFarm Research, "Neural Networks Documentation", from http://documents.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/2.5.1.html

[Num 2007] Visual Numerics, "IMSL® C Numerical Stat Library", 2007, http://www.vni.com/products/imsl/documentation/CNL06/stat/NetHelp/default.htm?turl=multilayerfeedforwardneuralnetworks.htm

[Lea 2008] eFunda, "Least Square Method", from http://www.efunda.com/math/leastsquares/leastsquares.cfm

[Erd 05] D. Erdogmus, O. Fontenla-Romero, J.C. Principe, A. Alonso-Betanzos, E. Castillo, "Linear-least-squares initialization of multilayer perceptrons through backpropagation of the desired response", *IEEE Trans. On Neural Networks*, vol.16, no.2, pp. 325-337, 2005.

[Liu 08] Y. Liu, “Hierarchical Self-Organizing Learning systems for Embodied Intelligence”, Ph.D. dissertation, Ohio University, Athens, OH, USA, 2008.

[Wikis] Wikipedia, the free encyclopedia. Available: http://en.wikipedia.org/wiki/Short-term_memory

[Wikil] Wikipedia, the free encyclopedia. Available: http://en.wikipedia.org/wiki/Long-term_memory

[Sta 2007] J. A. Starzyk, and H. He, “Anticipation-Based Temporal Sequences Learning in Hierarchical Structure”, *IEEE Trans. on Neural Networks, vol. 18, no. 2, pp. 344 – 358,* March 2007.

[Yuk 1999] D.S. Yuk, J. Flanagan, “Telephone Speech Recognition Using Neural Networks and Hidden Markov Model”, *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1999.

[Fal 1990] F. Fallside, H. Lucke, T.P. Marsland, Oapos, P.J. Shea, M.S.J. Owen, R. W. Prager, A.J. Robinson, N.H. Russell, “Continuous speech recognition for the TIMIT database using neural networks”, *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol.1 445-448, 1990.

[Gar 1993] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, V. Zue, “TIMIT Acoustic-Phonetic Continuous Speech Corpus”, *Linguistic Data Consortium*, Philadelphia, 1993.

APPENDIX A: SOURCE CODE AND DATA SETS USED IN THIS THESIS

The source code packages used in this thesis are available at the following address:

http://www.ent.ohiou.edu/~webcad/Current_Projects/Students_thesis_dissertation/Yiming

Source Code Packages:

Phoneme_Neural_Network: Package for the feed-forward phoneme recognition neural network

Sequence_Learning_LTM: Package for the sequence learning block composed by Long-Term Memory cells

Phoneme_Recognition_Feedback_Correction: Package for the phoneme recognition model which combine the feed-forward phoneme recognition neural network and the sequence learning block with feedback correction

These software packages can be provided upon request. Please send your request to: yh128999@ohio.edu . Each package includes a README file, please read it before use.

Data Sets:

In order to use the above software packages, the datasets used have to be from the TIMIT Acoustic-Phonetic Continuous Speech Corpus [Gar 1993].

To obtain TIMIT database, please visit the following address for ordering information:

http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1

APPENDIX B: HOW TO USE THE SOURCE CODE

Phoneme_Neural_Network

To use this phoneme neural network package, please follow the following steps:

1. Open SoundProcess.m

1.1 Change the filename to the file path where saves the training sets of the TIMIT database in your computer. (Make sure to change the names of the TIMIT files and folders to numbers so that the program can run in a loop to read all the data files.)

1.2 Choose the regions (indexi), the speakers (folders) and the sentences (files) for training.

1.3 Run this program. The outputs (A_matrix and B) of this program are saved into #.mat files in the working directory. Each file contains the outputs for one training sentence.

1.4 Change the file path to the testing sets of the TIMIT database, and choose the regions (indexi), the speakers (folders) and the sentences (files) for testing.

1.5 Uncomment line 116 and run the program again for testing sets. The outputs (A_matrix and B) of this program are saved into test#.mat files in the working directory. Each file contains the outputs for one testing sentence.

2. Open WordProcess.m

Use the same procedure as in step 1to obtain the preprocessed word information.

3. Open MLPProcessing.m

3.1 Change the path_data and test_path_data to your working directory or the directory saves the preprocessed sound information (.mat files).

3.2 Change the number of training sentences (file_index) and the number of testing sentences (test_file_index) as you want.

3.3 Run this program. This program is used to retrieve the preprocessed sound data from the saved .mat files and save them into large matrices for neural network to process.

4. Open main_MLP_LS.m (Important: must run MLPProcessing.m before this program)

4.1 Choose your desired number of layers (no_layer) and number of hidden neurons (B_neurons) in the neural network, and choose the transfer function you like to have on hidden neurons (nltype_hidden).

4.2 Run this program. The results are displayed in the console and also saved in train_error_result (the number of incorrect recognized phonemes in training set), train_error_rate (the error rate for the training data), test_error_result (the number of incorrect recognized phonemes in testing set), and test_error_rate (the error rate for the testing data).

Follow the above procedure with the following parameters to obtain the results shown in Table 1:

SoundProcess.m

When process training datasets:

Regions 1 – 8:	indexi = 1:8
Speakers 1 – 20:	folders = 1:20
Sample Files/Sentences 1 – 2:	samplefiles = 2

When process testing datasets:

Region 1 – 2:	indexi = 1:2
Speakers 1 – 10:	folders = 1:10
Sample Files/Sentences 1 – 2:	samplefiles = 2

WordProcess.m

Use the same parameters as in SoundProcess.m

MLPProcessing.m

Training Sentences 1 – 80:	file_index = 1:80
Testing Sentences 1 – 40:	test_file_index = 1:40

main_MLP_LS.m

Number of the Layers 3:	no_layer = 3
Hidden Neurons (50, 55, 60, 65, 70):	B_Neurons = 50, 55, 60, 65, 70
Hidden Neuron Transfer Function 2:	nltype_hidden = 2
Training Type 2:	training_type = 2

Sequence_Learning_LTM

To use this Sequence_Learning_LTM package, please follow the following steps:

1. Open LTMmain.m, change the decay rate as you desire, choose your own training text words and replace them in variable saved_text; choose your own testing text words and replace them in variable input_text.

2. Run the program. Variable strength1 saves the LTM responses for all the testing words.

Phoneme_Recognition_Feedback_Correction

To use this Phoneme Recognition with Feedback Correction package, please follow the following steps:

1. Open SoundProcess.m

    1.1 Change the filename to the file path where you save the training sets of the TIMIT database in your computer. (Make sure to change the names of the TIMIT files and folders to numbers so that the program can run in a loop to read all the data files.)

    1.2 Choose the regions (indexi), the speakers (folders) and the sentences (files) for training.

    1.3 Run this program. The outputs (A_matrix and B) of this program are saved into #.mat files in the working directory. One file contains the outputs for one training sentence.

1.4 Change the file path to the testing sets of the TIMIT database, and choose the regions (indexi), the speakers (folders) and the sentences (files) for testing.

1.5 Uncomment line 116 and run the program again for testing sets. The outputs (A_matrix and B) of this program are saved into test#.mat files in the working directory. One file contains the outputs for one testing sentence.

2. Open WordProcess.m

Use the same procedure as in step 1 above to obtain the preprocessed word information.

3. Open MLPProcessing.m

3.1 Change the path_data and test_path_data to your working directory or the directory saves the preprocessed sound information .mat files.

3.2 Choose the number of training sentences (file_index) and the number of testing sentences (test_file_index) as you want.

3.3 Run this program. This program is to retrieve the preprocessed sound data from the saved .mat files and save into large matrices for neural network to process.

4. Open main_MLP_LS.m (Important: must run MLPProcessing.m before this program)

4.1 Choose your desired number of layers (no_layer) and number of hidden neurons (B_neurons) in the neural network, and choose the tranfer function you like to have on hidden neurons (nltype_hidden).

4.2 Run this program. The results are displayed in the console and also saved in train_error_result (the number of incorrect recognized phonemes in training set), train_error_rate (the error rate for the training data), test_error_result (the number of incorrect recognized phonemes in testing set), and test_error_rate (the error rate for the testing data).

5. Open SequenceLearningFeedbackMain.m

5.1 Change the path_data and test_path_data to your working directory or the directory saves the preprocessed word sequence information dword_#.mat files.

5.2 Choose the number of training sentences (file_index) and the number of testing sentences (test_file_index) as you want.

5.3 Run this program. The results are displayed in the console and also saved in test_feedback_error_result (the number of incorrect recognized phonemes in testing set with feedback correction) and test_feedback_error_rate (the error rate for the testing data with feedback correction).

Follow the above procedure with the following parameters to obtain the results shown in Table 4:

SoundProcess.m

When process training datasets:

Regions 1 – 8:	indexi = 1:8
Speakers 1 – 20:	folders = 1:20
Sample Files/Sentences 1 – 2:	samplefiles = 2

When process testing datasets:

Region 1 – 2:	indexi = 1:2
Speakers 1 – 10:	folders = 1:10
Sample Files/Sentences 1 – 2:	samplefiles = 2

WordProcess.m

Use the same parameters as in SoundProcess.m

MLPProcessing.m

Training Sentences 1 – 80:	file_index = 1:80
Testing Sentences 1 – 40:	test_file_index = 1:40

main_MLP_LS.m

Number of the Layers 3:	no_layer = 3
Hidden Neurons (50, 55, 60, 65, 70):	B_Neurons = 50, 55, 60, 65, 70
Hidden Neuron Transfer Function 2:	nltype_hidden = 2
Training Type 2:	training_type = 2

SequenceLearningFeedbackMain.m

Training Sentences 1 – 80:	file_index = 1:80
Testing Sentences 1 – 40:	test_file_index = 1:40
Decay Rate 2:	decay = 2